



Guide to Developing with the GCSS-AF Integration Framework Appendix A

DRAFT

Prepared for:
Department of the Air Force
Headquarters Standard Systems Group (HQ SSG)
Maxwell Air Force Base - Gunter Annex
Montgomery, Alabama

Contract Number: **F01620-96-D-0004**

Document Number: **GCSS-REPORT-1997-0011 Appendix A**
Draft Version: **3.0**
Date: **03/23/01**



*-Lockheed Martin Systems Integration-Owego
1801 Route 17C
Owego, NY 13827-3998*



This Page Intentionally Left Blank.

DOCUMENT CHANGE HISTORY					
VERSION			CHANGE DESCRIPTION (REQUIRED FOR CHANGES AFFECTING TPM, REQUIREMENTS, CONFIGURATION, COST & SCHEDULE)		
NO.	APPROVAL	DATE	CHANGE DOC.	SECTION	NARRATIVE (OF ITEMS AFFECTED)
0.5		12/11/00			INITIAL DRAFT
1.0		1/12/01			RE-WRITE
1.5		02/21/01			RE-WRITE
2.0		03/09/01			DRAFT
2.5		03/16/01			DRAFT
3.0		03/23/01			DRAFT

This Page Intentionally Left Blank

TABLE OF CONTENTS

1.	APPENDIX A - UTILITY/HELPER CLASS AND API DESCRIPTIONS	1
1.1	INTRODUCTION.....	1
1.1.1	Purpose of this Document	1
1.1.2	Objectives.....	1
1.1.3	Scope.....	1
2.	CLASS AND API DESCRIPTIONS	3
2.1	TMESSAGE.....	3
2.2	COM.LMFS.FRAMEWORK.AMIHELPERS.*	9
2.3	COM.LMFS.FRAMEWORK.BOD.BOD (OAG BOD BASE CLASS).....	24
2.4	COM.LMFS.FRAMEWORK.PUBSUB.MQRFH - PUBLISH/SUBSCRIBE HELPERS	35
2.5	COM.LMFS.FRAMEWORK.SERVLET.IFSERVLET	46
2.6	COM.LMFS.FRAMEWORK.* - SERVLET UTILITY HELPERS	50
3.	SECURITY HELPERS	51
3.1	COM.LMFS.FRAMEWORK.LDAPHELPER.*	51
3.1.1	com.lmfs.framework.LDAPHelper.LDAPHelper	52
3.1.2	com.lmfs.framework.LDAPHelper.LDAPResultsHelper	53
3.1.3	com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator.....	55
3.1.4	com.lmfs.framework.LDAPHelper.LDAPTests	56
3.2	COM.LMFS.FRAMEWORK.SECURITY.*	56
3.2.1	com.lmfs.framework.security.IFSecurityException.....	57
3.2.2	com.lmfs.framework.security.IFMenuPOSCreator.....	57
3.2.3	com.lmfs.framework.security.PDCheckParser.....	58
3.2.4	com.lmfs.framework.security.PDCheckEnvironment.....	61
3.2.5	com.lmfs.framework.security.AuthInfo	62
3.3	COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.*.....	63
3.3.1	com.lmfs.framework.security.PDAuthn.PDAuthnInfo	64
3.3.2	com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl.....	64
3.3.3	com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl	66
3.3.4	com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl	67
3.3.5	com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl	67
3.4	COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.*	68
3.4.1	com.lmfs.framework.security.PDAuthz.PDCheckAuthz	69
3.4.2	com.lmfs.framework.security.PDAuthz.PDAuthorizedException	69
3.4.3	com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl.....	70
3.4.4	com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl	72
3.4.5	com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl	72
3.4.6	com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl	73
3.5	COM.LMFS.FRAMEWORK.UTIL.*	76
3.5.1	com.lmfs.framework.util.StringExt.....	76
3.5.2	com.lmfs.framework.util.ArrayExt	77
3.5.3	com.lmfs.framework.util.PropertiesExt	78
3.6	SESSIONINFO STRUCTURES.....	79
3.7	IFCBSECURITYINFO.....	81
3.7.1.1.1	Authentication.....	83
3.7.1.1.2	Providing Access Control Checks in MA Software	84

TABLE OF FIGURES

FIGURE 1: EXAMPLE OF CODE USING TMOUTBOUNDCOPYHELPER	4
FIGURE 2: EXAMPLE OF CODE USING THE TMINBOUND MESSAGE INTERFACE.....	5
FIGURE 3: CALLING ORDER OF SENDANDFORGET.JAVA	10
FIGURE 4: EXAMPLE CODE FROM THE FILEWRAPPER COMPONENT FOR SENDANDFORGET.JAVA	10
FIGURE 5: CALLING ORDER OF REQUESTRESPONSE.JAVA	11
FIGURE 6: EXAMPLE CODE EXCERPTED FROM THE FILEWRAPPER COMPONENT FOR REQUESTRESPONSE.JAVA	11
FIGURE 7: CALLING ORDER OF THE RECIEVER.JAVA.....	12
FIGURE 8: EXAMPLE CODE OF RECIEVER.JAVA.....	12
FIGURE 9: CALLING ORDER FOR PUBLISH.JAVA.....	13
FIGURE 10: EXAMPLE CODE FOR PUBLISH.JAVA	13
FIGURE 11: CALLING ORDER FOR SUBSCRIBE.JAVA.....	13
FIGURE 12: EXAMPLE CODE FOR SUBSCRIBE.JAVA	14
FIGURE 13: BUSINESS OBJECT DOCUMENT (BOD) ARCHITECTURE	25
FIGURE 14: AN EXAMPLE OF AN APPLICATION USING THE BASE BOD CLASS TO DETERMINE HOW TO HANDLE AN INCOMING UNSOLICITED MESSAGE:	27
FIGURE 15: EXAMPLE OF THE BOD CLASS USNG APPLICATION PROVIDED EXTENSIONS	31
FIGURE 16: EXAMPLE OF PUBLISH/SUBSCRIBE HELPER CODE	38
FIGURE 17: EXAMPLE THAT SHOWS USE OF PUBLISH/SUBSCRIBE JAVA BINDINGS	39
FIGURE 18 IFSERVLET EXTENDED CLASS DIAGRAM.....	47
FIGURE 19: LDAPHELPER EXAMPLE FROM MENU.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT	52
FIGURE 20: LDAPRESULTSHelper CODE EXAMPLE FROM THE MENU.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT	54
FIGURE 21: EXAMPLE OF PDCHECKPARSER CODE	59
FIGURE 22: EXAMPLE OF <IVCHECK> HTML TAG	59
FIGURE 23: EXAMPLES OF GETCLIENTIDENT, GETGROUPS, AND GETPAC FROM THE IFSERVLET GETSESSIONINFOSTRUCT METHOD.....	66
FIGURE 24: EXAMPLE OF CONSTRUCTING AND INITIALIZING A PDSEVRLETAUTHZIMPL OBJECT TAKEN FROM THE MENUSERVLET.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT.....	74
FIGURE 25: EXAMPLE OF PDCHECKAUTHORIZATIONPAC TAKEN FROM THE MENU.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT.....	75
FIGURE 26: EXAMPLE OF PROPERTIESEXT FROM THE MENUSERVLET.JAVA FILE OF THE MENU SYSTEM TEST COMPONENT	78
FIGURE 27: EXAMPLE OF USERSECINFOSTRUCT CODE	79
FIGURE 28: EXAMPLE OF APPSESSIONINFOSTRUCT CODE	80
FIGURE 29: EXAMPLE OF USERSESSIONINFOSTRUCT CODE.....	80
FIGURE 30: EXAMPLE OF SESSIONINFOSTRUCT CODE.....	80
FIGURE 31: IFCBSECURITYINFOSTRUCT	84
FIGURE 32: EXAMPLE IDL FROM PDCSESSIONMODULE OF THE PDC TEST COMPONENT	86
FIGURE 33: EXAMPLE IDL FROM COM_LMFS_FRAMEWORK_TESTCOMPONENTS_REQUSITION_ORDERINGTIE.JAVA OF THE REQUISITION COMPONENT TEST COMPONENT	87
FIGURE 34: EXAMPLE OF OBTAINING PARAMETERS REQUIRED BY THE GCSSACCESSDECISIONALLOWED METHOD	89
FIGURE 35: CODE EXAMPLE OF FINDING IFCBSECINFO HOME	90
FIGURE 36: CODE EXAMPLE OF CREATING IFCBSECURITYINFO INSTANCE	91
FIGURE 37: CODE EXAMPLE OF INVOKING THE GCSAFACCESSDECISIONALLOWED METHOD	92

TABLE OF TABLES

TABLE 1 INTEGRATION FRAMEWORK PROVIDED JAR FILE CONTENTS.....	2
TABLE 2 TMESSAGE	6
TABLE 3 COM.LMFS.FRAMEWORK.AMIHELPERS.*	14
TABLE 4 COM.LMFA.FRAMEWORK.BOD.BOD (BASE CLASS).....	32
TABLE 5 PUBSUBHELPERS (BINDINGS FOR C APPLICATIONS).....	40
TABLE 6 COM.LMFS.FRAMEWORK.SERVLET.IFSERVLET	48
TABLE 7 COM.LMFS.FRAMEWORK.* - SERVLET UTILITY HELPERS	50
TABLE 8 COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPHELPER	52
TABLE 9 COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPRESULTSHelper	54
TABLE 10: COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPMENUENTRYCOMPARATOR	55
TABLE 11 COM.LMFS.FRAMEWORK.LDAPHELPER.LDAPTESTS	56
TABLE 12 COM.LMFS.FRAMEWORK.SECURITY.IFSECURITYEXCEPTION	57
TABLE 13 COM.LMFS.FRAMEWORK.SECURITY.IFMENUPOSCREATOR	58
TABLE 14 COM.LMFS.FRAMEWORK.SECURITY.PDCHECKPARSER	60
TABLE 15 COM.LMFS.FRAMEWORK.SECURITY.PDCHECKENVIRONMENT	61
TABLE 16 COM.LMFS.FRAMEWORK.SECURITY.AUTHNINFO	63
TABLE 17 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.PDAUTHNINFO	64
TABLE 18 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.PDAUTHNINFOIMPL	65
TABLE 19 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.SERVLET.PDSERVLETAUTHNINFOIMPL	67
TABLE 20: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.APPLICATION.PDAPPAUTHNINFOIMPL	67
TABLE 21: COM.LMFS.FRAMEWORK.SECURITY.PDAUTHN.EJB.PDEJBAUTHNINFOIMPL	68
TABLE 22 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.PDCHECKAUTHZ.....	69
TABLE 23 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.PDAUTHZUNAUTHORIZEDEXCEPTION..	70
TABLE 24 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.PDCHECKAUTHZIMPL	71
TABLE 25 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.CB.PDCBAUTHZIMPL	72
TABLE 26 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.EJB.PDEJBAUTHZIMPL	73
TABLE 27 COM.LMFS.FRAMEWORK.SECURITY.PDAUTHZ.SERVLET.PDSERVLETAUTHZIMPL.....	75
TABLE 28 COM.LMFS.FRAMEWORK.UTIL CLASS	76
TABLE 29 COM.LMFS.FRAMEWORK.UTIL.STRINGEXT	76
TABLE 30 COM.LMFS.FRAMEWORK.UTIL.ARRAYEXT	77
TABLE 31 COM.LMFS.FRAMEWORK.UTIL.PROPERTIESEXT	78
TABLE 32: IFCBSECURITYINFO	81
TABLE 33: MA AUTHENTICATION SCENARIOS	83
TABLE 34: PARAMETERS OF THE GCSSAF/ACCESSDECISIONALLOWED METHOD	87

1. Appendix A - Utility/Helper Class and API Descriptions

1.1 Introduction

GCSS-AF provides a component-based Reference Architecture Framework that serves as the Integration and Application Framework Layers for GCSS-AF functional capabilities consistent with the Defense Information Infrastructure Common Operating Environment (DII COE), the Joint Technical Architecture - Air Force (JTA-AF), and based on commercial open standards. The GCSS-AF Reference Architecture Framework also provides common interfaces for those functions that either directly or indirectly support Command and Control (C2) or share information with C2 Systems.

It is assumed that the reader is familiar with the GCSS-AF Integration Framework and has at least a high level understanding of the services and facilities that it provides. This document is to be used as a reference regarding how to use those services and facilities.

1.1.1 Purpose of this Document

The purpose of this Appendix is to provide Application Developers with a description and examples of intended use of the helper and utility classes that are provided by the GCSS-AF Integration Framework (IF). This Appendix is also intended to direct developers to additional information required to develop applications employing the IF capabilities.

1.1.2 Objectives

The primary objective of this document is to provide the application developer with the information required in order to take advantage of the services that are provided by the GCSS-AF Integration Framework. Once familiar with the design guidance and application development approaches identified in the main body the [Guide to Developing with the GCSS-AF Integration Framework](#), an application developer should be able to use this appendix as an API dictionary and reference.

1.1.3 Scope

This document contains two main sections:

Section 2

This section contains the description of the utility and helper classes that are generally to be used during the development of Presentation and Business layer application components.

Section 3

This section contains the descriptions of the *utility* and *helper* classes that allow applications to make use of the security services provided by the IF.

Table 1: Integration Framework Provided jar File Contents identifies each of the utility and helper classes that are provided by the IF, the jar file that each is contained in, and a short

description of the intended purpose. Refer to the appropriate section of this appendix for additional details.

Table 1: Integration Framework Provided jar File Contents

Utility/Helper Package	Contained in.....	Short Description
TMMessage	N/A	Provides MQSeries communication support from within a business application deployed in the component broker.
com.lmfs.framework.amiHelpers.*	IFSServices.jar	A set of APIs that wrapper the OAMAS Application Messing Interface. These wrappers are used by applications executing outside the application server.
com.lmfs.framework.BOD.BOD (OAG BOD Base Class) The Open Application Group's (OAG) Business Object Document (BOD) is the architecture used to communicate messages or business documents between software applications or components. Each BOD includes supporting details to enable the destination <i>Business application</i> to accomplish the action. The BOD consists of two areas as shown in Figure 13 . • The Control Area • The Business Data Area.	IFSServices.jar	Base class provided by the Integration Framework that supports the building and parsing of OAG compliant BODs
com.lmfs.framework.pubsub.MQRFH - Publish/Subscribe Helpers	IFSServices.jar	Provides an API that encapsulates the MQSeries publish and subscribe message constructs and formats.
com.lmfs.framework.servlet.IFServlet	com_lmfs_framework_servlet.jar	A Servlet base class that provides the implementation of a set of services required by Servlets built using the Integration Framework, particularly security. This base class is extended by application developers to create application specific Servlets.
com.lmfs.framework.* - Servlet Utility Helpers	com_lmfs_framework.jar	Provides a set of classes to be used by applications for logging into the WAS-EE environment and for handling property file.
com.lmfs.framework_LDAPHelper.*	IFSServices.jar	Provides a set of classes that help simplify connecting to and interfacing with an LDAP repository.
com.lmfs.framework.security.*	IFSServices.jar	Provides a set of classes that may be used by applications to perform

Utility/Helper Package	Contained in.....	Short Description
		authorization checks and filtering of HTML documents.
com.lmfs.framework.security.PDAuthn.*	IFSServices.jar	Provides an interface for retrieving authentication information.
com.lmfs.framework.security.PDAuthz.*	IFSServices.jar	Provides an interface for authorization checks. This allows for an abstract implementation of the actual authorization engine to allow for ease of use within an application or Servlet
com.lmfs.framework.util.*	IFSServices.jar	A set of utility classes that extend the functionality of several standard classes provided by Java.
IFCFSecurityInfo	N/A	Provides application authentication and authorization support to EJB and CORBA components.

2. Class and API Descriptions

2.1 TMMessages

The **TextMessage** Component Broker application, also known at **TMMessages**, contains two classes, **TMOutbound** and **TMInbound**, that provide access to the attributes of messages (such as priority, encoding, and format) which are to be sent or received using MQSeries from within a Component Broker component. Note that in order to use this component in your server, your server in Component Broker must be configured in SMUI to use the **TextMessageApp**, **iDXAAAServices** and **iMQAAServices** applications.

When sending messages from an application the **TMOutboundCopyHelper** sets the desired attributes of the outgoing message. You would use the **TMOutboundCopyHelper**, for example, to set the queue to which you want to put the message, the message's priority, or the encoding of the message. The *copy* helper is then converted to a string and ultimately passed to the *Put* method of the WebSphere Enterprise Edition MQSeries Application Adapter (MQAA). The MQAA ensures that the message is placed on the specified queue.

The following is an example of Java code to send a message using this interface:

```
// copy for TMOutbound
TextMessageCopy.TMOutboundCopy outCopy = TextMessageCopy.TMOutboundCopyHelper._create() ;

// set destination queueName
outCopy.queueName( destQueue ) ;

// set message data
outCopy.messageData( msgByteArray ) ;

// set encoding
outCopy.encoding( MQC.MQENC_INTEGER_NORMAL | MQC.MQENC_DECIMAL_NORMAL |
MQC.MQENC_FLOAT_IEEE_NORMAL ) ;

// set format to be a simple character string message
outCopy.format(MQC.MQFMT_STRING) ;

// convert the copy helper to a byte array
theCopyOrKeyStr = outCopy._toString() ;

// put the message to the output queue.
getOutboundMsgHome().put(theCopyOrKeyStr) ;
```

Figure 1: Example of Code Using TMOutboundCopyHelper

When receiving messages, the **TMInbound** interface contains the characteristics of the message as specified by the sender of the message. The receiver may require some of these attributes in order to respond to a request contained in the message or to confirm the receipt of the message.

The following is an example of code using the **TMInbound** message interface.

```
// string version of message text
String msgString = null ;
// inbound message object
TextMessage.TMInbound inMsg = null ;

// byte array for key or copy helper use
byte theCopyOrKeyStr[] = null ;

// key for TMInbound
TextMessageKey.TMInboundKey inKey =
TextMessageKey.TMInboundKeyHelper._create() ;
// set queue name to access for message
inKey.queueName(queueToGetFrom) ;

// get the Key in byte[]
theCopyOrKeyStr = inKey._toString() ;

// get the ExtendedInboundMessageQueue
IExtendedMessagingModule.ExtendedInboundMessageQueue inHome =
IExtendedMessagingModule.ExtendedInboundMessageQueueHelper.narrow(getInboundMsgHome()) ;

try {

    // begin transaction
    currentTransaction.begin();

    // get the message
    obj = inHome.getUsingKeyString(theCopyOrKeyStr) ;
    // No message in the queue
    currentTransaction.commit(true) ;
} // end catch IMessageNotFound
catch (Exception exc) {
    // commit the transaction
    currentTransaction.commit(true) ;
    // No message in the queue
    currentTransaction.commit(true) ;
} // end catch IMessageNotFound
catch (Exception exc) {
    // commit the transaction
    currentTransaction.commit(true) ;
} // end catch Exception
```

Figure 2: Example of Code Using the TMInbound Message Interface

Table 2: TMMessages

TMMessages		
TMOOutbound		
replyToQ	• N/A	The string that specifies the name of the queue, which the recipient of the message is to send the requested reply. If there is no expected response to the message, this field remains blank.
replyToQMgr	• N/A	The string that specifies the name of the queue manager where the <i>reply</i> queue exists. If this field remains blank, the queue manager shall fill it in before placing the message on the specified queue. The name used shall be the same as the queue manager the sending application is using.
queueName	• N/A	The string that specifies the name of the queue to which the message is to be put.
messageData	• N/A	The bytestring that contains the message itself. This is the sent data.
disposition	• N/A	The long that contains the disposition option, which can be specified as part of the report options of the MQMD structure. It can be set to either MQRO_DEAD_LETTER_Q (the default) or MQRO_DISCARD_MSG . Further information on this attribute can be obtained from the Description of the MQMD structure in the MQSeries Application Programming Reference. (*MQSAPR)¹
msgType	• N/A	The long that contains the message type field of the MQMD structure. *MQSAPR
expiry	• N/A	The long that specifies the length of time, which the message be allowed to remain on the recipients' queue. Once this time-period has expired the message shall be erased or moved to the deadletter Queue .
encoding	• N/A	The long that identifies the representation used for numeric values in the application message data; this applies to binary integer data, packed-decimal integer data, and floating-point data.
codedCharSetId	• N/A	The long that specifies the <i>coded character set identifier</i> of character data in the application message data.
format	• N/A	The string that specifies the format of the message data. Two examples of the field content are MQC.MQFMT_STRING for <i>string</i> type messages and MQFMT_RF_HEADER for <i>published</i> messages.
priority	• N/A	The long that specifies the priority of the message.
persistence	• N/A	The long that specifies the persistence of the message. *MQSAPR

¹ Further information on this attribute can be obtained from the Description of the MQMD structure in the *MQSeries Application Programming Reference. (*MQSAPR)*

TMMessages		
msgId	• N/A	The string that is used to distinguish one message from another. No two messages <i>should</i> have the same message identifier, although this is allowed by the queue manager. The message identifier is a permanent property of the message, and shall remain after restarts of the queue manager.
correlId	• N/A	The StringCorrelationID . This field correlates a response to a particular request. This field set to the value contained in the incoming request, from the TMInbound message value.
putApplType	• N/A	The long that is the type of application, which put the message. This is part of the origin context of the message. *MQSAPG
putApplName	• N/A	The string that is part of the origin context of the message. The format of the name depends on the PutApplType . *MQSAPG
putDate	• N/A	The string that is the date when the message was put . This is part of the origin context of the message. *MQSAPG
putTime	• N/A	The string that is the time when the message was put . This is part of the origin context of the message. *MQSAPG
applOriginData	• N/A	The string that is <i>Application data</i> relating to origin. This is part of the origin context of the message. *MQSAPG
TMInbound		
replyToQ	• N/A	The string that specifies the name of the queue where the <i>response</i> message is placed. This field remains empty unless the incoming message is a request or the sender expects confirmation of receipt.
replyToQMgr	• N/A	The string that specifies the name of the queue manager where the <i>reply</i> queue is hosted.
queueName	• N/A	The string that specifies the name of the queue to which the message is to be read.
messageData	• N/A	The bytestring that contains the message data, which was retrieved from the MQSeries queue.
msgType	• N/A	The long that contains the message type field of the MQMD structure. *MQSAPR
expiry	• N/A	The long that specifies the length of time, which the message shall be allowed to remain on the recipients' queue. Once this time-period has expired the message shall be erased or moved to the deadletter queue.
encoding	• N/A	The long that identifies the representation used for numeric values in the application message data;

TMMessages		
		this applies to binary integer data, packed-decimal integer data, and floating-point data.
codedCharSetId	• N/A	The long that specifies the coded character set identifier of character data in the application message data.
priority	• N/A	The long that specifies the priority of the message.
persistence	• N/A	The long that specifies the persistence of the message. *MQSAPR
putApplType	• N/A	The long that is the type of application, which put the message. This is part of the origin context of the message. *MQSAPG
msgId	• N/A	The string that is used to distinguish one message from another. No two messages <i>should</i> have the same message identifier, although this is allowed by the queue manager. The message identifier is a permanent property of the message, and shall remain after restarts of the queue manager.
correlId	• N/A	The string that is the <i>correlation ID</i> . This field correlates a response to a particular request.
qMgrName	• N/A	The string that specifies the name of the queue manager to which the message shall be put.
putApplName	• N/A	The string that is part of the origin context of the message. The format of the name depends on the PutApplType . *MQSAPG
putDate	• N/A	The string that is the date when the message was put. This is part of the origin context of the message. *MQSAPG
putTime	• N/A	The string that is the time when the message was put. This is part of the origin context of the message. *MQSAPG
applOriginData	• N/A	The string that is Application data relating to origin. This is part of the origin context of the message. *MQSAPG .
waitInterval	• N/A	The long that specifies how long to wait for an incoming message before timing out.
backoutCount	• N/A	The long attribute that specifies the number of times, which a message was retrieved from the queue from within a transaction that was rolled back. This value determines the number of attempted retries on any given message.
feedback	• N/A	The long that is a feedback or reason code. This works with a message type MQMT_REPORT to indicate the nature of the report, and is only meaningful with that type of message. *MQSAPR
userIdentifier	• N/A	The string that is part of the identity context of

TMMessages		
		the message; it identifies the user who originated the message. The queue manager treats this information as character data, but does not define the format of it. *MQSAPR
accountingToken	• N/A	The string that is part of the identity context of the message; it allows an application to cause work done as a result of the message to be appropriately charged.
applIdentityData	• N/A	The string that is application data relating to identity. This is part of the identity context of the message; it is information that the application suite defines. It provides additional information about the message or its originator.
format	• N/A	The string that specifies the format of the incoming message data. MQC.MQFMT_STRING for "string-type" messages and MQFMT_RF_HEADER for published messages are examples of the contents of this field.
stripRFHeader	• None	A method that returns a bytestring , which is the message data without the publish/subscribe header attached. If the application does not need the information contained in the publish/subscribe header, this method is used. The message shall be treated like a <i>string</i> type message.

2.2 com.lmfs.framework.amiHelpers.*

The **OAMAS API** is the specification for the messaging interface between the business components to which the framework adheres. It provides separation of *application business functions* from the *infrastructure support*. The Integration Framework AMI helper and template classes encapsulate the Vendor's *OAMAS API* implementation. These are API wrappers on top of the Message-Oriented Middleware Application Messaging Interface. These *helper* classes constitute the interface, which application developer's use for applications meant to execute outside the application server. This makes the task of the application developer simpler and it removes the specifics of the product from the interface.

The AMI helper classes maintain an internal state that includes connections with one or more queue managers. This internal state is initialized with the **create()** call and connection(s) to MQSeries queue manager is initialized with the **open()** call. The **close()** call disconnects from the queue manager and releases any resources that were allocated by **open()** or **create()**. AMI uses policies to control many aspects of the connection to MQSeries and the processing of **puts** and **gets** to queues. Please see the AMI documentation for a thorough description of policies and their use.

There are five AMI helper classes. These classes support the various messaging styles for the sender and receiver roles. Messaging systems support multiple styles of communication between applications that include *send and forget*, *request/reply*, and *publish/subscribe*. To obtain a complete *OAMAS API* specification and messaging style descriptions, visit the **Open Applications Group** website at: <http://www.openapplications.org/oamas/loadform.htm>.

An application using the *send and forget* style sends a message to another application or list of applications but does not expect any reply. An example is a *Data Replication Application*.

Applications using *request/reply* are like client server applications where a client is making a request from a server and expecting a reply.

Publish/Subscribe

Publish/subscribe is similar to *send and forget* except that the sending application does not know who the recipients are. Instead, the sender sends the message to a broker who manages the subscriptions of applications that requested to *receive* messages. The broker decides which subscribing applications should receive a published message by matching the subscriptions with either the message topic information or the content of the message.

SendAndForget.java

This class simplifies the sending of messages using the *send and forget* messaging style.

The methods of this class are called in the following order:

```
create(...)  
open()  
send(...)  
  
.  
  
.  
  
send(...)  
close()
```

Figure 3: Calling Order of SendAndForget.java

Example code from the **Filewrapper** component:

```
private static com.lmfs.framework.amiHelpers.SendAndForget sender;  
    sender = new com.lmfs.framework.amiHelpers.SendAndForget();  
    sender.create("GCSS.IF.IFSTC3FILEWRAPPER.SESSION",  
"GCSS.IF.DEFAULT.POLICY", args[5],  
        "GCSS.IF.IFSTC3FILEWRAPPER.SEND.MESSAGE");  
    sender.open();  
    sender.setRetryCount(3);  
    sender.send(theMessage);  
    (sender.close() not performed because file wrapper loops forever and reuses sender.)
```

Figure 4: Example Code from the Filewrapper Component for SendAndForget.java

RequestResponse.java

This class simplifies the sending and receipt of messages using the *request/reply* messaging style.

The methods of this class are called in one of the following order:

```
create(...)  
open()  
receiveRequest()  
sendReply(...)
```

```
.
```

```
.
```

```
receiveRequest()  
sendReply(...)  
close()
```

Or

```
create(...)  
open()  
sendRequest()  
receiveReply(...)
```

```
.
```

```
sendRequest()  
receiveReply(...)  
close()
```

Figure 5: Calling Order of RequestResponse.java

Example code excerpted from the **Filewrapper** component:

```
private static com.lmfs.framework.amiHelpers.RequestResponse sendAndReceiver;  
    sendAndReceiver = new com.lmfs.framework.amiHelpers.RequestResponse();  
    sendAndReceiver.create("GCSS.IF.IFSTC3FILEWRAPPER.SESSION",  
"GCSS.IF.DEFAULT.POLICY",  
        args[5], args[6],  
"GCSS.IF.IFSTC3FILEWRAPPER.SEND.MESSAGE",  
        "GCSS.IF.IFSTC3FILEWRAPPER.RECEIVE.MESSAGE");  
    sendAndReceiver.open();  
    sendAndReceiver.setRetryCount(3);  
    // send the request  
    sendAndReceiver.sendRequest(theMessage);  
    ...  
    sendAndReceiver.setWaitTime(60000);  
    theReply = sendAndReceiver.receiveReply();  
(sendAndReceiver.close() not performed because file wrapper loops forever and reuses  
sendAndReceiver.)
```

Figure 6: Example Code Excerpted from the Filewrapper Component for RequestResponse.java

Receiver.java

This class simplifies the receipt of messages. The **Receiver.java**, along with the *send and forget* class, can be used to implement the *send and forget* messaging style or *the request/reply* messaging style. The *request/response* class is also provided.

The methods of this class are called in the following order:

```
private static com.lmfs.framework.amiHelpers.RequestResponse sendAndReceiver;  
    sendAndReceiver = new com.lmfs.framework.amiHelpers.RequestResponse();  
    sendAndReceiver.create("GCSS.IF.IFSTC3FILEWRAPPER SESSION",  
    "GCSS.IF.DEFAULT.POLICY",  
        args[5], args[6],  
    "GCSS.IF.IFSTC3FILEWRAPPER SEND MESSAGE",  
        "GCSS.IF.IFSTC3FILEWRAPPER RECEIVE MESSAGE");  
    sendAndReceiver.open();  
    sendAndReceiver.setRetryCount(3);  
    // send the request  
    sendAndReceiver.sendRequest(theMessage);  
    ...  
    sendAndReceiver.setWaitTime(60000);  
    theReply = sendAndReceiver.receiveReply();  
(sendAndReceiver.close() not performed because file wrapper loops forever and reuses  
sendAndReceiver.)
```

Figure 7: Calling Order of the Reciever.java

Example code:

```
Receiver r = new Receiver();  
r.create("AMT.SAMPLE.SESSION", "AMT.SAMPLE.POLICY",  
"AMT.SAMPLE.RECEIVER",  
    "AMT.SAMPLE.MESSAGE.NAME");  
r.open();  
String msg = r.receive();  
System.out.println("received message: " + msg);  
r.close();
```

Figure 8: Example Code of Reciever.java

Publisher.java

This class simplifies the publication of string format messages on a given topic. **Publisher.java**, along with the *subscribe* class, implements the *publish/subscribe* messaging style.

This class creates the objects needed to *publish* a message, opens them for use, publishes a message, and receives the confirm message from the broker.

The methods are called in this order:

```
create(...)  
open()  
publish(...)  
. . .  
publish(...)  
close()
```

Figure 9: Calling Order for Publish.java

Example code:

```
Publisher p = new Publisher();  
p.create("AMT.SAMPLE.SESSION", "AMT.SAMPLE.POLICY",  
"AMT.SAMPLE.PUBLISHER",  
        "AMT.SAMPLE.RECEIVER", "AMT.SAMPLE.MESSAGE1.NAME",  
        "AMT.SAMPLE.MESSAGE1.NAME");  
p.open();  
p.publish("topic we are publishing on", "message we are publishing");  
p.close();
```

Figure 10: Example Code for Publish.java

Subscriber.java

This class simplifies the management of subscriptions and the receipt of messages that are published to topics that have been subscribed to. **Subscriber.java** along with the *publish* class, can be used to implement the *publish/subscribe* messaging style.

The methods of this class are called in the following order:

```
create(...)  
open()  
subscribe(...)  
receive()  
. . .  
receive()  
unsubscribe(...)  
close()
```

Figure 11: Calling Order for Subscribe.java

Example code:

```
Subscriber s = new Subscriber();
s.create("AMT.SAMPLE.SESSION", "AMT.SAMPLE.POLICY",
"AMT.SAMPLE.SUBSCRIBER",
        "AMT.SAMPLE.SENDMESSAGE.NAME", "AMT.SAMPLE.GETMESSAGE.NAME");
s.open();

s.subscribe("topic we are subscribing to");
String msg = s.receive();
System.out.println("received message: " + msg);
s.close();
```

Figure 12: Example Code for Subscribe.java

Table 3: com.lmfs.framework.amiHelpers.*

Com.lmfs.framework.amiHelpers.*	
Class AMIHelperObject	
Public void setRetryCount(int value)	
• value - (int) Specifies the number of retries attempted for a given action. (publish, send, receive, etc.)	This method allows the number of retries for sending and receiving messages to be set to a given value.
Public int getRetryCount()	
• Returns int - The current value of the retry count.	This method allows an application to retrieve the current value of the retry count.
Public void enableWarnings(boolean arg)	
• arg - (boolean) true enable AMI warning exceptions; false disables them.	Allows and application to enable (arg=true) or disable (arg=false) AMI warning exceptions.
Public void clearErrorCodes() throws AmErrorException, AmWarningException	
• Throws AmErrorException –A generic MQSeries AMI error exception.	Clears error codes in the AMI policy object.
• Throws AmWarningException –A generic MQSeries AMI warning error exception.	
Public AmStatus getLastErrorHandler() throws AmErrorException, AmWarningException	
• Returns AmStatus – The status of the last error condition from the AMI policy object.	Returns the AmStatus of the last error condition from the AMI policy object.
• Throws AmErrorException – A generic MQSeries AMI error exception.	
• Throws AmWarningException –A generic MQSeries AMI warning error exception.	
Public void create(String sessionName, String policyName) throws AmErrorException, AmWarningException	
• SessionName - (String) The caller-selected name given to the session.	Common code for the create() operation: This method creates the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object. This method is called by the create() methods of all the AMI helper classes.
• PolicyName - (String) The name of the AMI policy to use.	
• Throws AmErrorException –A generic MQSeries AMI error exception.	
• Throws AmWarningException –A generic MQSeries AMI warning error exception.	

Com.lmfs.framework.amiHelpers.*	
Public void open() throws AmWarningException, AmErrorException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> • Throws AmErrorException – A generic MQSeries AMI error exception. • Throws AmWarningException – A generic MQSeries AMI warning error exception. • Throws AMIHelperExceptionQueueManagerNotRunning – An exception that indicates the queue manager may not be running. • Throws AMIHelperExceptionQueueDoesNotExist –An exception that indicates the target queue may not exist. 	Common code for the open() operation: Open session with retries. This method initializes the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object. This method is called by the open() methods of all the AMI helper classes.
Public void close() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> • Throws AmErrorException –A generic MQSeries AMI error exception. • Throws AmWarningException – A generic MQSeries AMI warning error exception. 	Common code for close operation: Close session. This method closes the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object. This method is called by the close() methods of all the AMI helper classes.
Public void setWaitTime(int waitTime) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> • WaitTime - (int) new wait time (in milliseconds). • Throws AmErrorException –A generic MQSeries AMI error exception. • Throws AmWarningException –A generic MQSeries AMI warning error exception. 	<i>Setter</i> for wait time in the policy of this application.
Public int getWaitTime() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> • Returns int – The current value of the wait time from the Policy in use. • Throws AmErrorException –A generic MQSeries AMI error exception. • Throws AmWarningException – A generic MQSeries AMI warning error exception. 	<i>Getter</i> for wait time from the policy for this application.
Public void beginTransaction() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException - A generic MQSeries AMI warning error exception. 	This is the method you would call to begin a MQSeries unit of work, or “transaction.” You would also need to check the “Syncpoint” box in the “General” tab of the policy you plan to use. You would check the box inside the AMI Administration tool.
Public void commitTransaction() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException - A generic MQSeries AMI warning error exception. 	This is the method you would call to commit a MQSeries unit of work, or “transaction.” You would also need to check the “Syncpoint” box in the “General” tab of the policy you plan to use. You would check the box inside the AMI Administration tool.
Public void rollbackTransaction() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> • Throws AmErrorException - A generic MQSeries AMI error exception 	This is the method you would call to rollback a MQSeries unit of work, or “transaction.” You would also need to

Com.lmfs.framework.amiHelpers.*	
<ul style="list-style-type: none"> Throws AmWarningException - A generic MQSeries AMI warning error exception. 	check the “Syncpoint” box in the “General” tab of the policy you plan to use. You would check the box inside the AMI Administration tool.
Class Publisher	
	Public void create(String sessionName, String policyName, String publisherName, String responseName, String messageName, String respName) throws AmErrorException, AmWarningException
<ul style="list-style-type: none"> SessionName - (String) AMI session name chosen by the application developer. PolicyName - (String) AMI policy name defined through the AMI administration console. PublisherName - (String) AMI publisher service point name defined through the AMI administration console. ResponseName - (String) AMI receiver service point name for the <i>confirm message</i> which the broker shall send to the application after a <i>publish()</i>. MessageName - (String) AMI message name for the message to publish. This is a unique name chosen by the application. RespName - (String) AMI message name for the <i>confirm message</i> the application shall receive from the broker. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Creates all of the internal objects of the AMI helper class needed to <i>publish</i> a message and receive the subsequent <i>confirm message</i> from the broker.
Public void open() throws AMIHelperExceptionQueueManagerNotRunning, AmErrorException, AmWarningException, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionQueueManagerNotRunning –An exception that indicates the queue manager may not be running. Throws AMIHelperExceptionQueueDoesNotExist –An exception that indicates the target queue may not exist. 	Opens all of the objects needed. This method initializes the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object.
Public void publish(String topic, String message) throws AmErrorException, AmWarningException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> Topic - (String) The topic on which to publish the message. Message - (String) The message to publish. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws 	Publishes the contents of a Java string using the specified topic. Also receives and discards the confirmation message from the broker.

Com.lmfs.framework.amiHelpers.*	
<p>AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no confirm message could be found on the receiver service point</i>.</p> <ul style="list-style-type: none"> Throws AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the confirm message is <i>get inhibited</i>. Throws AMIHelperExceptionQueueIsPutInhibited – Indicates that the underlying queue for the publisher is <i>put inhibited</i>. Throws AMIHelperExceptionQueueManagerNotRunning –An exception that indicates the queue manager may not be running. 	
public void close() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Closes all of the AMI MQSeries objects used internally by this class
class Receiver	
public void create(String sessionName, String policyName, String servicePointName, String messageName) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> SessionName - (String) AMI session name. PolicyName - (String) AMI policy name. ServicePointName - (String) AMI receiver service point name. MessageName - (String) AMI message name for incoming message. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Creates all of the objects needed to send a message0 .
public void open() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueDoesNotExist, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionQueueDoesNotExist –An exception that indicates the queue manager may not be running. Throws AMIHelperExceptionQueueManagerNotRunning - Exception that indicates the target queue may not exist. 	Opens all of the objects needed. This method initializes the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object.
public String receive() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> Returns String – String containing text of received message. 	Receives a message and returns it in string form to the caller

Com.lmfs.framework.amiHelpers.*	
<p>received message.</p> <ul style="list-style-type: none"> • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException – A generic MQSeries AMI warning error exception. • Throws AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the incoming message is <i>get</i> inhibited. • Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver</i> service point. • Throws AMIHelperExceptionQueueManagerNotRunning – An exception that indicates the queue manager may not be running. 	
<p>public String browse() throws AmErrorException, AmWarningException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning</p>	
<ul style="list-style-type: none"> • Returns String – String containing text of browsed message. • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException - A generic MQSeries AMI warning error exception. • Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver</i> service point. • Throws AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the incoming message is <i>get</i> inhibited. • Throws AMIHelperExceptionQueueManagerIsNotRunning – An exception that indicates the queue manager may not be running. 	Browses a message and returns it in string form to the caller. Unlike receive , the message remains in the underlying message queue.
<p>public void close() throws AmErrorException, AmWarningException</p>	
<ul style="list-style-type: none"> • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Closes all of the AMI MQSeries objects used by this class
class RequestResponse	
<p>public void create(String sessionName, String policyName, String senderName, String receiverName, String sendMessageName, String receiveMessageName) throws AmWarningException, AmErrorException</p>	
<ul style="list-style-type: none"> • SessionName - (String) AMI session name. • PolicyName - (String) AMI policy name. • SenderName - (String) AMI sender service point name. • ReceiverName - (String) AMI receiver service point name. This is where the response is 	Creates all of the objects needed to send and receive requests and replies

Com.lmfs.framework.amiHelpers.*	
<ul style="list-style-type: none"> expected to arrive. SendMessageName - (String) AMI message name of outgoing message. ReceiveMessageName - (String) AMI message name of incoming message. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	
public void open() throws AmWarningException, AmErrorException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionQueueManagerNotRunning - Exception that indicates the queue manager may not be running. Throws AMIHelperExceptionQueueDoesNotExist - An exception that indicates the target queue may not exist. 	Opens all of the objects needed. This method initializes the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object.
public void openAsResponder() throws AmWarningException, AmErrorException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. Throws AMIHelperExceptionQueueDoesNotExist - An exception that indicates the target queue may not exist. 	Opens all of the objects needed to act as a responder. This method initializes the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object when acting as a responder.
public String receiveRequest() throws AmWarningException, AmErrorException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> Returns String – String containing request message text. Throws AmErrorAn exception - A generic MQSeries AMI error An exception. Throws AmWarningAn exception - A generic MQSeries AMI warning error An exception. Throws AMIHelperAn exceptionNoMesageOnQueue – Indicates <i>no message</i> on the queue underlying receiver service point. Throws AMIHelperAn exceptionQueueIsGetInhibited – Indicates that 	Receive an incoming request message

Com.lmfs.framework.amiHelpers.*	
<p>the queue underlying receiver service point is <i>get</i> inhibited.</p> <ul style="list-style-type: none"> • Throws AMIHelperAn exceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	
public void sendReply(String reply) throws AmWarningException, AmErrorException, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> • reply - (String) The string containing the <i>reply</i> message text. • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException - A generic MQSeries AMI warning error exception. • Throws AMIHelperExceptionQueueIsPutInhibited – Indicates that the underlying queue for the sender is <i>put</i> inhibited. • Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	Sends a <i>reply</i> to the previous request
public void sendRequest(String request) throws AmWarningException, AmErrorException, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> • Request - (String) The string containing the message text to be sent. • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException - A generic MQSeries AMI warning error exception. • Throws AMIHelperExceptionQueueIsPutInhibited – Indicates that the underlying queue for the sender is <i>put</i> inhibited. • Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	Sends a request to which a <i>reply</i> is expected
public String receiveReply() Throws AmWarningException, AmErrorException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> • Returns String – String containing the contents of the <i>reply</i> message. • Throws AmErrorException - A generic MQSeries AMI error exception. • Throws AmWarningException - A generic MQSeries AMI warning error exception. • Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver service point</i>. 	Receive the <i>reply</i> to the previous request

Com.lmfs.framework.amiHelpers.*	
<ul style="list-style-type: none"> Throws – AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the receive message is get inhibited. Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	
<pre>public String receiveUncorrelatedReply() throws AmWarningException, AmErrorException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning</pre>	
<ul style="list-style-type: none"> Returns String – String containing the contents of the received message. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException – A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver service point</i>. Throws AMIHelperExceptionQueueIsGetInhibited – Indicates that the underlying queue for the receive message is <i>get</i> inhibited. Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	Receives the next message on the <i>receive</i> queue, is not correlated with any previous request
<pre>public void close() throws AmWarningException, AmErrorException</pre>	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Closes all of the AMI MQSeries objects used internally by this class.
<pre>public void handleErrors(AmErrorException errorEx) throws AmWarningException, AmErrorException</pre>	
<ul style="list-style-type: none"> eErrorEx - (AmErrorException) The exception to be handled. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Process errors encountered during an MQSeries send or receive operation
class SendAndForget	
<pre>public void create(String sessionName, String policyName, String senderName, String messageName) throws AmErrorException, AmWarningException</pre>	
<ul style="list-style-type: none"> SessionName - (String) AMI session name. PolicyName - (String) AMI policy name. SenderId - (String) AMI sender service point name. MessageName - (String) AMI message name for outgoing message. Throws AmErrorException - A generic 	Creates all of the internal state objects of this AMI helper needed for sending messages.

Com.lmfs.framework.amiHelpers.*	
MQSeries AMI error exception. <ul style="list-style-type: none"> Throws AmWarningException - A generic MQSeries AMI warning error exception. 	
public void open() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueManagerNotRunning, AMIHelperExceptionQueueDoesNotExist	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. Throws AMIHelperExceptionQueueDoesNotExist - An exception that indicates the target queue may not exist. 	Opens all of the objects needed. This method initializes the AmSession and AmPolicy objects that are part of the internal state of the AMI helper object.
public void send(String message) throws AmErrorException, AmWarningException, AMIHelperExceptionQueueIsPutInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> message - (String) String containing message text to be sent. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionQueueIsPutInhibited - Indicates that the underlying queue for the sender is <i>put</i> inhibited. Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	Sends a datagram built from the Message String Input Parameter
public void close() throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Closes all of the AMI MQSeries objects used by this class
class Subscriber	
public void create(String sessionName, String policyName, String subscriberName, String sendMsgName, String receiveMsgName) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> SessionName - (String) AMI session name. PolicyName - (String) AMI policy name. SubscriberName - (String) AMI subscriber service point name. sendMsgName - (String) AMI message name for the published message. ReceiveMsgName - (String) AMI message name for incoming message. Throws AmErrorException - A generic MQSeries AMI error exception. 	Creates all of the internal objects of the AMI helper class needed to subscribe to a topic and <i>receive</i> a published message.

Com.lmfs.framework.amiHelpers.*	
<ul style="list-style-type: none"> Throws AmWarningException - A generic MQSeries AMI warning error exception. 	
public void open() throws AmErrorException, AmWarningException, AMIHelperExceptionQueueDoesNotExist, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionQueueDoesNotExist – An exception that indicates the target queue may not exist. Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	Opens all of the objects needed. This method initializes the AmSession and AmPolicy objects that are part of the internal state of the AI helper object.
public void subscribe(String topic) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> topic - (String) The <i>subscribe</i> topic. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	<i>Subscribes to a topic</i>
public String receive() throws AmErrorException, AmWarningException, AMIHelperExceptionNoMessageOnQueue, AMIHelperExceptionQueueIsGetInhibited, AMIHelperExceptionQueueManagerNotRunning	
<ul style="list-style-type: none"> Returns String – String containing text of received message. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. Throws AMIHelperExceptionNoMessageOnQueue – Indicates the <i>no message could be found on the receiver</i> service point. Throws AMIHelperExceptionQueueIsGetInhibited- Indicates that the underlying queue for the incoming message is <i>get</i> inhibited. Throws AMIHelperExceptionQueueManagerNotRunning - An exception that indicates the queue manager may not be running. 	<i>Receives a published message and returns it in string form to the caller</i>
public void unsubscribe(String topic) throws AmErrorException, AmWarningException	
<ul style="list-style-type: none"> topic - (String) The topic from which to <i>unsubscribe</i>. Throws AmErrorException - A generic MQSeries AMI error exception. Throws AmWarningException - A generic MQSeries AMI warning error exception. 	Removes the subscription to a given topic
Public void close() throws AmErrorException, AmWarningException	

Com.lmfs.framework.amiHelpers.*	
<ul style="list-style-type: none">• Throws AmErrorException - A generic MQSeries AMI error exception. <p>Throws AmWarningException - A generic MQSeries AMI warning error exception.</p>	Closes all of the AMI MQSeries objects used by this class

2.3 com.lmfs.framework.BOD.BOD (OAG BOD Base Class)

The Open Application Group's (OAG) **Business Object Document (BOD)** is the architecture used to communicate messages or business documents between software applications or components. Each BOD includes supporting details to enable the destination *Business application* to accomplish the action.

The BOD consists of two areas as shown in Figure 13.

- The Control Area
- The Business Data Area.

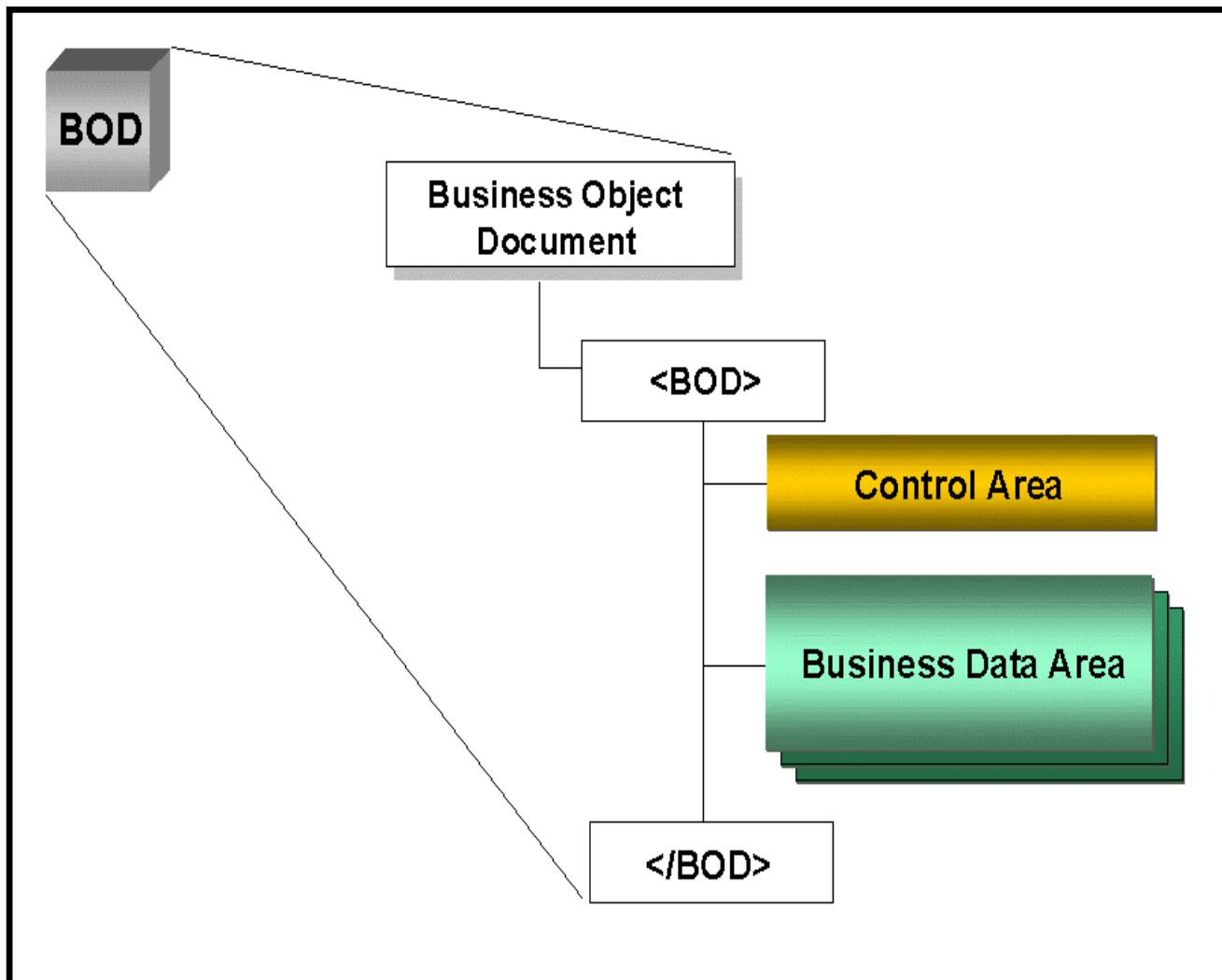


Figure 13: Business Object Document (BOD) Architecture

Provided with the IF is the BOD base class (**BOD.java**) as well as the application specific BODs which are extensions of that class used by the various provided test components. Several extended classes to the basic BOD class are included as examples for application developer use.

The base BOD class consists of the methods used to create, parse, and access the basic elements of the XML document. The *getter* and *setter* methods for elements in the *Control Area* of every BOD are in this class. The extension classes contain the *getter* and *setter* methods for each element in each of the other specific BODs.

The following example of an application uses the base BOD class to determine how to handle an incoming unsolicited message:

```
try { // BOD Handling
    // Reset the flag for confirm BOD - if generic BOD parser fails, do not send a response
    iBODConfirm = "" ;

    // Reset the flag for BOD Description
    iBODDescr = "" ;

    // Create a generic BOD
    BOD theBOD = new BOD(msgString);

    if ( logCat.isDebugEnabled() )

        logCat.debug("PDCSessionAO- " + location() + "::messageReady::Generic BOD parsed
successfully") ;

    // save the Description of the BOD
    iBODDescr = theBOD.getVERB() + theBOD.getNOUN() + theBOD.getREVISION() ;

    if ( (iBODDescr.equals("SYNCINVENTORY003")) ) { // SyncInventoryBOD
        if ( logCat.isDebugEnabled() )
            logCat.debug("PDCSessionAO- " + location() + "::messageReady::SyncInventoryBOD
received") ;

        // save the Control Area of the BOD
        iBODCtrlArea = theBOD.getCNTROLAREA() ;

        // save the Confirmation Flag
        iBODConfirm = theBOD.getCONFIRMATION() ;

        // SyncInventory BOD from EPD Wrapper to Enterprise PDC
        // parse and process the message content - invoke handleSyncInventoryBOD
        replyMsgString = handleSyncInventoryBOD(msgString) ;

        // send the reply if requested - ConfirmBOD
        if ( (iBODConfirm.equals("1")) || (iBODConfirm.equals("2")) ) {

            sendMessage(inMsg.replyToQ(),
                        inMsg.correlator(),
                        replyMsgString.getBytes(),
                        MQC.MQFMT_STRING ) ;

        }
    }
}
```

```
        } // end if
    } // end if SyncInventoryBOD

    if ( logCat.isDebugEnabled() )
        logCat.debug("PDCSessionAO-" + location() + "::messageReady::Current
transaction committed" );

} // end try - BOD handling
catch (BODEException be) {
    if ( iErrorMessage.equals("") )
        iErrorMessage = "Cannot parse the generic BOD, bad file passed" ;
    // Log Error
    logCat.error("PDCSessionAO-" + location() + "::messageReady::" + iErrorMessage,
be) ;
    throw be ;
} // end catch BODEException
```

Figure 14: An Example of an Application using the Base BOD Class to Determine How to Handle an Incoming Unsolicited Message:

The following is an example of the BOD class using application provided extensions:

```
private java.lang.String handleSyncInventoryBOD( java.lang.String theSyncInvBOD)
throws com.ibm.IManagedClient.IDuplicateKey,
PDCHelperModule.PDCEception
{
// <GeneratedMethodBody>
// <Body origin="user" xmi.uuid="DCE:96E70DA4-6808-11d4-8B62-000629059EDD:1">
// Insert Method modifications here
/**
 * This method shall handle a SyncInventoryBOD. It shall parse all the
 * data areas out of the BOD and process them.
 * @return java.lang.String
 * @param java.lang.String theSyncInvBOD
 * @exception com.ibm.IManagedClient.IDuplicateKey
 * @exception PDCHelperModule.PDCEception
 */

if ( logCat.isDebugEnabled() )
logCat.debug("Entering PDCSessionAO-" + location() + "::handleSyncInventoryBOD") ;

// local variables
//      return message string - Confirm BOD
String confirmStr = null ;
//      reference to pdc
PDCModule.PartsDataCollection thePDC = null ;

try {
// get the reference to PDC
thePDC = findPDCByPrimaryKey(1) ;
}
catch (com.ibm.IManagedClient.INoObjectWKey nowk) {
logCat.error("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Cannot find
reference to PDC!!", nowk) ;
}

try { // Parse and process BOD
// create the SyncInventoryBOD from the string
SynInventoryBOD theBOD = new SynInventoryBOD(theSyncInvBOD) ;
if ( logCat.isDebugEnabled() )
logCat.debug("PDCSessionAO-" + location() +
)::handleSyncInventoryBOD::SyncInventoryBOD parsed successfully") ;

// save the Control Area of the BOD
iBODCtrlArea = theBOD.getCNTROLAREA() ;
// save the Confirmation Flag
iBODConfirm = theBOD.getCONFIRMATION() ;

// process the BOD
// get the number of data areas
int numOfDataAreas = theBOD.getNumOfDATAAREA() ;

for(int i = 0; i < numOfDataAreas; i+=1) {
// move to the current Data Area and initialize
```

```
theBOD.moveToDATAAREA(i) ;
    theBOD.initDataAreapointers() ;
    NDC.push("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Current Data Area = " +
theBOD.getCurrentLocationOfDATAAREA() ;

    // create a PartRecord
    PDCHelperModule.PartRecord pRecord = new PDCHelperModule.PartRecord() ;

    // get the values
    // stock Number
    pRecord.stockNumber = theBOD.getITEM() ;
    if ( logCat.isDebugEnabled() )
        logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Stock Number is: " +
pRecord.stockNumber) ;

    // part Number
    pRecord.partNumber = theBOD.getPARTNUM() ;
    // item Description
    pRecord.itemDescription = theBOD.getDESCRIPTN() ;
    // manufacturer
    pRecord.manufacturer = theBOD.getMANUFACTR() ;
    // quantity
    String itmQtyNumOfDec = theBOD.getITEMQTYNUMOFDEC() ;
    String itmQtySign = theBOD.getITEMQTYSIGN() ;
    String itmQtyUOM = theBOD.getITEMQTYUOM() ;

    String itmQtyVal = null ;
    int itmQty = 0 ;

    if ( (itmQtyNumOfDec.equals("0")) && (itmQtySign.equals("+")) && ( (itmQtyUOM.equals("EACH"))
|| (itmQtyUOM.equals(" ")) ) ) {
        itmQtyVal = theBOD.getITEMQTYVALUE() ;
        itmQty = new Integer(itmQtyVal).intValue() ;
    }
    else if ( (itmQtyNumOfDec.equals("0")) && (itmQtySign.equals("+")) &&
(itmQtyUOM.equals("DOZEN")) ) {
        itmQtyVal = theBOD.getITEMQTYVALUE() ;
        itmQty = new Integer(itmQtyVal).intValue() ;
        itmQty *= 12 ;
    }
    pRecord.quantity = itmQty ;

    // security Label
    //pRecord.secLabel = theBOD.getSECLABEL();

    // determine what to do based on the action code.
    if (theBOD.getSYNCIND().equals("A")) {
        if ( logCat.isDebugEnabled() )
            logCat.debug("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Add a part from
SyncInvBOD") ;
        try {
            thePDC.addPart( pRecord ) ;
            iStatusLvl = "00" ;
            iReasonCd = "Message: SyncInventoryBOD-Add was processed successfully" ;
        }
```

```
if ( logCat.isDebugEnabled() )
    logCat.debug("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::" + iReasonCd) ;
}
catch (com.ibm.IManagedClient.IDuplicateKey idk) {
    // Set the Error Message
    iErrorMessage = "Duplicate Key, Part with StockNumber: " + pRecord.stockNumber + " already
exists!!";
    iStatusLvl = "99";
    iReasonCd = iErrorMessage;
    logCat.warn("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::" + iErrorMessage, idk)
;
    throw idk;
} // end catch idk - A
} // end if "A"
else if(theBOD.getSYNCIND().equals("C")) {
    if ( logCat.isDebugEnabled() )
        logCat.debug("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::Update a part from
SyncInvBOD");
    try {
        thePDC.updatePart( pRecord );
        iStatusLvl = "00";
        iReasonCd = "Message: SyncInventoryBOD-Update was processed successfully";
        if ( logCat.isDebugEnabled() )
            logCat.debug("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::" + iReasonCd);
    }
    catch (com.ibm.IManagedClient.INoObjectWKey nk) {
        // Set the Error Message
        iErrorMessage = "Updating Part with StockNumber: " + pRecord.stockNumber + " does not exist!!"
;
        iStatusLvl = "99";
        iReasonCd = iErrorMessage;
        logCat.warn("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::" + iErrorMessage, nk)
;
    } // end catch nk - C
} // end else if "C"
else if(theBOD.getSYNCIND().equals("D")) {
    if ( logCat.isDebugEnabled() )
        logCat.debug("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::Remove a part from
SyncInvBOD");
    try {
        thePDC.removePart( pRecord.stockNumber );
        iStatusLvl = "00";
        iReasonCd = "Message: SyncInventoryBOD-Delete was processed successfully";
        if ( logCat.isDebugEnabled() )
            logCat.debug("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::" + iReasonCd);
    }
    catch (com.ibm.IManagedClient.INoObjectWKey nk) {
        // Set the Error Message
        iErrorMessage = "Removing Part with StockNumber: " + pRecord.stockNumber + " does not
exist!!";
        iStatusLvl = "99";
        iReasonCd = iErrorMessage;
        logCat.warn("PDCSessionAO- " + location() + "::handleSyncInventoryBOD::" + iErrorMessage, nk)
;
    } // end catch nk - D
}
```

```
NDC.pop();
} // end for
} // end try Parse and process BOD
catch (BODEException be) {
// Set the Error Message
iErrorMessage = "Error on parsing the SyncInventoryBOD" ;
logCat.error("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::" + iErrorMessage, be)
;
throw new PDCHelperModule.PDCEception( be.getMessage() ) ;
} // end catch be

// Build the Confirm BOD if required
try {
if ( (iBODConfirm.equals("1")) || (iBODConfirm.equals("2")) ) { // Send Confirm message
requested
// Build the Reply BOD - ConfirmBOD
ConfirmBOD confBOD = new ConfirmBOD();

// set ConfirmBOD's values
confBOD.setSTATUSLVL(iStatusLvl) ;
confBOD.setREASONCODE(iReasonCd) ;
confBOD.setDATAAREACNTROLAREA(iBODCtrlArea) ;
confBOD.setDESCRIPTN(iBODDescr) ;

// Get the String Version of the ConfirmBOD
confirmStr = confBOD.getBOD() ;
} // end if
}
catch (Exception exc) {
logCat.error("PDCSessionAO-" + location() + "::handleSyncInventoryBOD::Exception caught
(ConfirmBOD): ", exc);
throw new PDCHelperModule.PDCEception( exc.getMessage() );
}

if ( logCat.isDebugEnabled() )
logCat.debug("Exiting PDCSessionAO-" + location() + "::handleSyncInventoryBOD") ;

return confirmStr ;

// End Method modifications here
// </Body>
// </GeneratedMethodBody>
}
```

Figure 15: Example of the BOD Class Using Application Provided Extensions

Refer to the *Open Applications Group Integration Specification* (OAGIS) for any information needed concerning BOD. Complete description of the structure of a BOD, the data elements that are contained in the control area, the data elements that have been defined and accepted for use with in BODs, and the allowable contents of a data element. The specification is available from the Open Applications Group's website at: <http://www.openapplications.org>.

Table 4: com.lmfa.framework.BOD.BOD (Base Class)

com.lmfs.framework.BOD.BOD (Base Class)	
Public BOD()	
<ul style="list-style-type: none"> • N/A 	Constructor creates a new BOD object. This creates the document root, attaches a fully populated CNTROLAREA and a stub for the DATAAREA
Public BOD(String xmlBOD) throws BODException	
<ul style="list-style-type: none"> • XmlBOD - (String) An XML file (represented as a String) containing a BOD. • Throws BODException - When the SAX Parse fails. 	Creates and parses a new BOD from an existing XML file
Public void addDATAAREA()	
<ul style="list-style-type: none"> • N/A 	Adds another DATAAREA to the BOD and make the new DATAAREA the current DATAAREA
public void AddDateTimeQual(String qualifier, Element node)	
<ul style="list-style-type: none"> • qualifier - (String) Specifies the date time variety. For example, is this a date and time for Creation, delivery date, etc. • node - (Element) The created element in the BOD document. 	Appends the datetime segment with the requested qualifier to the requested element, setting default time parameters. For example to add a datetime segment such as Creation date/time to a node such as ReqLnHdr in a BOD, you would use this method and specify the qualifier Creation and the node ReqLnHdr . The result would be the Creation date/time segment being appended to the ReqLnHdr node of the created BOD.
public void AddDateTimeQual(String qualifier, Element node, String yearSt, String monthSt, String daySt, String hourSt, String minSt, String secSt, String subsecSt, String tzSt)	
<ul style="list-style-type: none"> • qualifier - (String) Specifies the date time variety. For example, is this a date and time for Creation, delivery date, etc. • node - (Element) The element in the created BOD document. • yearSt - (String) The year to be used in the date field. • monthSt - (String) The month to be used in the date field. • daySt - (String) The day of the month to be used in the date field. • hourSt - (String) The hour of the day to be used in the date field. • minSt - (String) The minute within the hour to be used in the date field. • secSt - (String) The second within the minute to be used in the date field. • subsecSt - (String) The fractional part of the second to be used in the date field. • tzSt - (String) The timezone to be used in the date field. 	Appends the date/time segment with the requested qualifier to the requested element, setting the passed in time parameters. For example, to add a date/time segment such as Creation date/time to a node such as ReqLnHdr in a BOD, you would use this method and specify the qualifier Creation , and the node ReqLnHdr . The result would be the Creation date/time segment would get appended to the ReqLnHdr node of the BOD being created. This method allows you to set the date/time fields at creation time.
public void BODToFile(String file)	
<ul style="list-style-type: none"> • file - (String) <i>File Name</i> of file to be written to. 	Prints to a file to form an XML document. A ".xml" extension shall automatically be appended This is a utility function only
public void BODtoScreen()	
<ul style="list-style-type: none"> • N/A 	Prints the XML file representing the BOD to the screen.

com.lmfs.framework.BOD.BOD (Base Class)	
	This is a utility function only
public String getAUTHID()	
• Returns String - The AUTHID.	Returns the AUTHID of the BOD
public String getBOD()	
• Returns String - The XML document.	Converts the XML document representing the BOD to a String and return the String
public String getCNTROLAREA()	
• Returns String - The CNTRLAREA.	Returns the entire CNTRLAREA of the BOD
public String getCNTROLAREAdata()	
• Returns String - The data contained in the CNTRLAREA.	Retrieves the data contained in the CNTROLAREA without any formatting or tags
public String getCodepage()	
• Returns String – The value of the CODEPAGE element.	Retrieves the value for the CODEPAGE
public String getCOMPONENT()	
• Returns String - The value of the COMPONENT element.	Retrieves the value for the COMPONENT
public String getCONFIRMATION()	
• Returns String - The value of the CONFIRMATION field.	Retrieves the CONFIRMATION field value
• 0 - Do NOT send a CONFIRM BOD	
• 1 – Send a CONFIRM BOD only if there is an error.	
• 2 - Send a CONFIRM BOD	
public String getDAY()	
• Returns String - The value of the DAY element.	Retrieves the value for DAY
public String getHOUR()	
• Returns String - The value of the HOUR element.	Retrieves the value for HOUR
public String getLANGUAGE()	
• Returns String - The value of the LANGUAGE element.	Retrieves the value for the LANGUAGE
public String getLOGICALID()	
• Returns String - The value of the LOGICALID element.	Retrieves the value for the LOGICALID
public String getMINUTE()	
• Returns String - The value of the MINUTE element.	Retrieves the value for the MINUTE
public String getMONTH()	
• Returns String – The value of the MONTH element.	Retrieves the value for the MONTH
public String getNOUN()	
• Returns String - The value of the NOUN element.	Retrieves the value for the NOUN
public int getNumOfDATAAREA()	
• Returns int – The number of DATAAREA's.	Retrieves the number of DATAAREAs
public String getREFERENCEID()	
• Returns String - The value of the REFERENCEID element.	Retrieves the value for the REFERENCEID

com.lmfs.framework.BOD.BOD (Base Class)	
public String getREVISION()	
• Returns String - The value of the REVISION element.	Retrieves the value for the REVISION
public String getSECOND()	
• Returns String – The value of the SECOND element.	Retrieves the value for the SECOND
public String getSUBSECOND()	
• Returns String – The value of the SUBSECOND element.	Retrieves the value for the SUBSECOND
public String getTASK()	
• Returns String - The value of the TASK element.	Retrieves the value for the TASK
public String getTimezone()	
• Returns String - The value of the TIMEZONE element.	Retrieves the value for the TIMEZONE
public String getVERB()	
• Returns String – The value of the VERB element.	Retrieves the value for the VERB
public String getYEAR()	
• Returns String - The value of the YEAR element.	Retrieves the value for the YEAR
public void moveToDATAAREA(int danum)	
• danum - (int) Which DATAAREA node to make current.	Move to the specified DATAAREA Note: the first one is zero. SHALL BE A VALID INDEX
public void moveToNextDATAAREA()	
• N/A	Moves to the next DATAAREA in the BOD
public void moveToPreviousDATAAREA()	
• N/A	Moves to the previous DATAAREA in the BOD
public void setAUTHID(String value)	
• Value - (String) The value of the AUTHID field.	Sets the AUTHID field
public void setCODEPAGE(String value)	
• value - (String) The value of the CODEPAGE field.	Sets the CODEPAGE field
public void setCOMPONENT(String value)	
• value - (String) The value of the COMPONENT field.	Sets the COMPONENT field
public void setCONFIRMATION(String value)	
• value - (String) The value of the CONFIRMATION field: <ul style="list-style-type: none"> • 0 - Do NOT send a CONFIRM BOD • 1 - Send a CONFIRM BOD only if there is an error • 2 - Send a CONFIRM BOD 	Sets the CONFIRMATION field
public void setLANGUAGE(String value)	
• value - (String) The value of the LANGUAGE field.	Sets the LANGUAGE field
public void setLOGICALID(String value)	
• value - (String) The value of the LOGICALID field.	Sets the LOGICALID field

com.lmfs.framework.BOD.BOD (Base Class)	
public void setNOUN(String value)	
• value - (String) The value of the NOUN field.	Sets the NOUN field
public void setREFERENCEID(String value)	
• value - (String) The value of the REFERENCEID field.	Sets the REFERENCEID field
public void setREVISION(String value)	
• value - (String) The value of the REVISION field.	Sets the REVISION field
public void setTASK(String value)	
• value - (String) The value of the TASK field.	Sets the TASK field
public void setVERB(String value)	
• value - (String) The value of the VERB field.	Sets the VERB field

2.4 com.lmfs.framework.pubsub.MQRFH - Publish/Subscribe Helpers

The *publish/subscribe helper* classes encapsulate the MQSeries *publish* and *subscribe* message constructs and formats. When an application uses the *publish/subscribe* messaging style, the application communicates with the *publish/subscribe* broker of MQSeries to accomplish various tasks such as registering or de-registering a publisher or subscriber of information of a particular topic. The broker requires specially formatted messages for these tasks. The purpose of the *publish/subscribe helper* classes is to simplify the construction of the broker control messages. The application developer calls a method, which puts the formatted broker control message into a buffer that can then be forwarded to the broker.

An example of how the *Trigger Monitor Application* template provided with the Integration Framework uses the *publish/subscribe helper* bindings for C applications follows below:

This code reads lines from a file and looks for stanzas beginning with [**registerPublisher**], [**registerSubscriber**], [**deregisterPublisher**], and [**deregisterSubscriber**]. When it finds one of these tokens, it reads in additional input lines that are to be used as parameters when the **publish/subscribe** helper classes are to be invoked. Then the code calls the helper class, which corresponds to the stanza it is processing. The helper class formats a buffer with the message needed to instruct the broker to perform the given operation. Then the code sends the message to the broker using **putMsg()**.

Example:

```
while( !read_line(inputFile, myLine, sizeof(myLine)) ) {
    memset(msg, 0, sizeof(msg));

    if ( !strcmp(myLine, "[registerPublisher]" ) ) {

        debug( "found registerPublisher" );

        read_line(inputFile, exceptions, sizeof(exceptions));
        debug( "exceptions = ", exceptions );
        if (find_exception(exceptions)) {
            debug( "found exception ... skipping to next stanza" );
            break;
        }

        read_line(inputFile, topic, sizeof(topic));
        read_line(inputFile, stream, sizeof(stream));
        expand_loc(stream);
        read_line(inputFile, qmgr, sizeof(qmgr));
        read_line(inputFile, queue, sizeof(queue));
        expand_loc(queue);
        read_line(inputFile, broker_queue, sizeof(broker_queue));
        read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

        debug( "topic = ", topic );
        debug( "stream = ", stream );
        debug( "qmgr = ", qmgr );
        debug( "queue = ", queue );
        debug( "broker_queue = ", broker_queue );
        debug( "broker_qmgr = ", broker_qmgr );

registerPublisher(msg,
    topic, // topic to subscribe to
    stream, // stream to subscribe to
    qmgr, // qmgr where reply comes to
    queue); // queue name where reply comes to

        putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

    } // if

    if ( !strcmp(myLine, "[registerSubscriber]" ) ) {
        debug( "found registerSubscriber" );

        read_line(inputFile, exceptions, sizeof(exceptions));
        debug( "exceptions = ", exceptions );
        if (find_exception(exceptions)) {
            debug( "found exception ... skipping to next stanza" );
            break;
        }
    }
}
```

```
read_line(inputFile, topic, sizeof(topic));
    read_line(inputFile, stream, sizeof(stream));
    expand_loc(stream);
    read_line(inputFile, qmgr, sizeof(qmgr));
    read_line(inputFile, queue, sizeof(queue));
    expand_loc(queue);
    read_line(inputFile, broker_queue, sizeof(broker_queue));
    read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

    debug( "topic = ", topic );
    debug( "stream = ", stream );
    debug( "qmgr = ", qmgr );
    debug( "queue = ", queue );
    debug( "broker_queue = ", broker_queue );
    debug( "broker_qmgr = ", broker_qmgr );

registerSubscriber(msg,
    topic, // topic to subscribe to
    stream, // stream to subscribe to
    qmgr, // qmgr where subscribed messages shall arrive
    queue); // queue name where subscribed messages shall

arrive

    putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

} // if

if ( !strcmp(myLine, "[deregisterPublisher]" ) {

    debug( "found deregisterPublisher" );

    read_line(inputFile, exceptions, sizeof(exceptions));
    debug( "exceptions = ", exceptions );
    if (find_exception(exceptions)) {
        debug( "found exception ... skipping to next stanza" );
        break;
    }

    read_line(inputFile, topic, sizeof(topic));
    read_line(inputFile, stream, sizeof(stream));
    expand_loc(stream);
    read_line(inputFile, qmgr, sizeof(qmgr));
    read_line(inputFile, queue, sizeof(queue));
    expand_loc(queue);
    read_line(inputFile, broker_queue, sizeof(broker_queue));
    read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

    debug( "topic = ", topic );
    debug( "stream = ", stream );
    debug( "qmgr = ", qmgr );
    debug( "queue = ", queue );
    debug( "broker_queue = ", broker_queue );
    debug( "broker_qmgr = ", broker_qmgr );
```

```
deregisterPublisher(msg,
    topic, // topic to subscribe to
    stream, // stream to subscribe to
    qmgr, // qmgr where reply comes to
    queue); // queue name where reply comes to

putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

} // if

if ( !strcmp(myLine, "[deregisterSubscriber]" ) ) {

    debug( "found deregisterSubscriber" );

    read_line(inputFile, exceptions, sizeof(exceptions));
    debug( "exceptions = ", exceptions );
    if (find_exception(exceptions)) {
        debug( "found exception ... skipping to next stanza" );
        break;
    }

    read_line(inputFile, topic, sizeof(topic));
    read_line(inputFile, stream, sizeof(stream));
    expand_loc(stream);
    read_line(inputFile, qmgr, sizeof(qmgr));
    read_line(inputFile, queue, sizeof(queue));
    expand_loc(queue);
    read_line(inputFile, broker_queue, sizeof(broker_queue));
    read_line(inputFile, broker_qmgr, sizeof(broker_qmgr));

    debug( "topic = ", topic );
    debug( "stream = ", stream );
    debug( "qmgr = ", qmgr );
    debug( "queue = ", queue );
    debug( "broker_queue = ", broker_queue );
    debug( "broker_qmgr = ", broker_qmgr );

    deregisterSubscriber(msg,
        topic, // topic to subscribe to
        stream, // stream to subscribe to
        qmgr, // qmgr where reply comes to
        queue); // queue name where reply comes to

    putMsg (broker_qmgr, broker_queue, (((PMQRFH)msg)->StrucLength), msg );

} // if

} // while
```

Figure 16: Example of Publish/Subscribe Helper Code

The following example of code demonstrates how a Java application would then use the **publish/subscribe** Java bindings to publish information to subscribers of a topic. In this example a buffer is created which contains a message to the broker instructing it to publish the data contained in **replyMsgString** using the topic **SYNCINVENTORY**. The buffer is then sent to the broker using **sendMessage()**.

```
// build the SyncInventoryBOD
String replyMsgString = buildSyncInventoryBOD( pRecord, "D" ) ;

// set the format variable

// Publish the remove to the BASEs that subscribe
// publish the change
// need String topic, String qname, String qmname,
// String puboptions, byte[] buffer
byte[] replyByteArray = mqps.publishWithData( "SYNCINVENTORY",
    "",
    "",
    "",
    replyMsgString ) ;

// set the destination queueName to be the Stream queue
// set to IF.<location>.DEFAULT.STREAM
// the broker shall put the SyncInventory BOD from Enterprise PDC (Publisher)
// to the Base 1, 2, 3 PDC's (Subscribers) whose queues are named as follows:
// PDC.<location>.SYNCINVENTORY.RECEIVER where loc = BASE1, BASE2, or
BASE3
String destQueue = "IF." + nameContext + ".DEFAULT.STREAM" ;
// set the correlator to be a empty string
String correlator = "" ;

// start a transaction
// obtain access to a transaction control object.
obj = CBSeriesGlobal.orb().resolve_initial_references("TransactionCurrent") ;
currentTransaction = org.omg.CosTransactions.CurrentHelper.narrow(obj) ;
currentTransaction.set_timeout(180) ;
if ( logCat.isDebugEnabled() )
    logCat.debug("PDCSessionAO- " + location() + "::removePart::Beginning current
transaction (Outbound)" );
// begin transaction
currentTransaction.begin() ;

// Send the message
sendMessage( destQueue,
    correlator,
    replyByteArray,
    MQFMT_RF_HEADER ) ;

// end transaction
currentTransaction.commit(true) ;
```

Figure 17: Example that Shows Use of publish/subscribe Java bindings

Table 5: PubSubHelpers (Bindings for C Applications)

Pub-Sub Helpers (Bindings for C Applications)	
void buildMQRFHeader(PMQRFH pRFHeader)	This function builds the RFH which is used for all publish/subscribe message
<ul style="list-style-type: none"> PRFHeader - (PMQRFH) A pointer to the region of pre-allocated memory in which the <i>MQRFHeader</i> is to be constructed. 	
void deletePublication (void *pMsgBuffer, char *pTopic, char *pStream)	This function builds NameValueString to delete a retained publication
<ul style="list-style-type: none"> pMsgBuffer - (void *) A pre-allocated buffer that returns the completed <i>name/value pair</i> string pTopic - (char *) The topic of the publication to be deleted. If none is provided, all shall be deleted for that stream. PStream - (char *) The topic stream for which the publication shall be deleted. If none is provided the default stream shall be used. 	
void deregisterPublisher (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	This function builds NameValueString to indicate that a publisher no longer publishes data for the topic indicated
<ul style="list-style-type: none"> pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string. pTopic - (char *) The topic of the publication to be deleted. A topic shall be provided. pStream - (char *) The topic stream for the topics named. If none is provided the default stream is used. pQMgrName - (char *) The queue manager named on the registerPublisher call. If none is provided the values from the message descriptor shall be used. pQName - (char *) The queue named on the registerPublisher call. If none is provided the values from the message descriptor shall be used. 	
void deregisterSubscriber (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	This function builds NameValueString to indicate that a subscriber shall no longer receive data for the topic indicated
<ul style="list-style-type: none"> pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string. pTopic - (char *) The topic of the publication to be deleted. A topic shall be provided. pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used. pQMgrName - (char *) The queue manager named on the <i>registerSubscriber</i> call. If none is provided the values from the message descriptor shall be used. pQName - (char *) The queue named on the registerSubscriber call. If none is provided the values from the message descriptor shall be used. 	
void publish (void *pMsgBuffer, char *pTopic, char *pQMgrName, char *pQName, char *pPublishOptions)	This function builds NameValueString to publish a
<ul style="list-style-type: none"> pMsgBuffer - (void *) A pre-allocated buffer 	

Pub-Sub Helpers (Bindings for C Applications)	
which shall be used to return the completed <i>name/value pair</i> string.	publication
<ul style="list-style-type: none"> • pTopic - (char *) The topic of the publication. A topic shall be provided. • pQMgrName - (char *) The queue manager named on the <i>registerPublisher</i> call. If none is provided the values from the message descriptor shall be used. • pQName - (char *) The queue named on the <i>registerPublisher</i> call. If none is provided the values from the message descriptor shall be used. • pPublishOptions - (char *) MQPS_RETAIN_PUBLICATION or none. 	
void registerPublisher (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> • pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string. • pTopic - (char *) The topic of the publication to be published. A topic shall be provided. • pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used. • pQMgrName - (char *) The queue manager to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used. • pQName - (char *) The queue name to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used. 	This function builds NameValueString to indicate that a publisher shall publish data for the topic indicated
void registerSubscriber (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> • pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string. • pTopic - (char *) The topic of the publication to be deleted. A topic shall be provided. • pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used. • pQMgrName - (char *) Name of queue manager to which messages shall be sent. If none is provided the values from the message descriptor shall be used. • pQName - (char *) Name of queue to which messages shall be sent. If none is provided the values from the message descriptor shall be used. 	This function builds NameValueString to indicate that a subscriber shall receive data for the topic indicated
void requestUpdate (void *pMsgBuffer, char *pTopic, char *pStream, char *pQMgrName, char *pQName)	
<ul style="list-style-type: none"> • pMsgBuffer - (void *) A pre-allocated buffer which shall be used to return the completed <i>name/value pair</i> string. 	This function builds NameValueString to request retained publications for the topic indicated

Pub-Sub Helpers (Bindings for C Applications)	
<ul style="list-style-type: none"> • pTopic - (char *) The topic of the publications requested. • pStream - (char *) The topic stream for the topics named. If none is provided the default stream shall be used. • pQMgrName - (char *) The queue manager to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used. • pQName - (char *) The queue name to which confirmation messages are to be sent. If none is provided the values from the message descriptor shall be used. 	
com.lmfs.framework.pubsub.MQRFH - Pub-Sub Helpers (Bindings for Java Applications)	
public void reset() throws Exception	
<ul style="list-style-type: none"> • Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown. 	This method sets all RFH attributes to default values.
public void readIn(MQMessage msg) throws Exception	
<ul style="list-style-type: none"> • msg - (MQMessage) The message object to parse. • Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown. 	Parses the RFH and NameValueString from an RFH format MQSeries message object
public void padNameValuePair() throws Exception	
<ul style="list-style-type: none"> • Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown. 	Adds white space to ensure full <i>Word Boundary Alignment</i> .
public void addNameValuePair (String name, String value) throws Exception	
<ul style="list-style-type: none"> • name - (String) Name to append. • value - (String) Value to append. • Throws Exception-Any exceptions thrown by the underlying string classes are re-thrown. 	Concatenates the name and value strings passed to the global NameValueString .
public void writeOut(MQMessage msg) throws Exception	
<ul style="list-style-type: none"> • msg - (MQMessage) The message object to build. • Throws Exception- Any exceptions thrown by the underlying string classes are re-thrown. 	Builds a MQMessage data buffer using the RFH header and global NameValueString .
public int registerPublisher(String topic, String stream, String qname, String qmname, byte[] buffer) throws Exception	
<ul style="list-style-type: none"> • topic - (String) The topic to register. • stream - (String) The stream to register. • qname - (String) The queue to which confirmations are to be sent. • qmname - (String) The queue manager. to which confirmations are to be sent. • buffer - (byte[]) The buffer in which the <i>namevalue</i> string is stored. • Returns int – The length of the <i>publish/subscribe</i> header. • Throws Exception- Any exceptions thrown by the underlying string classes are re-thrown. An exception is thrown if the topic is not supplied. 	This function builds NameValueString to indicate that a publisher shall publish data for the topic indicated

Pub-Sub Helpers (Bindings for C Applications)	
public void registerPublisher(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> topic - (String) The topic to register stream - (String) The stream to register qname - (String) The queue to which confirmations are to be sent. qmname - (String) The queue manager to which confirmations are to be sent. msg - (MQMessage) The message object in which the name value string is stored. Throws Exception- Any exceptions thrown by the underlying <i>string</i> classes are re-thrown. An exception is thrown if the topic is not supplied. 	This function builds NameValueString to indicate that a publisher shall publish data for the topic indicated
Public int deletePublication (String topic, String stream, byte[] buffer) throws Exception	
<ul style="list-style-type: none"> topic - (String) Topic of publication to delete. stream - (String) Stream of publication to delete. buffer - (byte[]) The buffer in which the name value string is stored. Returns int – The length of the publish/subscribe header. Throws Exception- - Any exceptions thrown by the underlying <i>string</i> classes are re-thrown. 	This function builds NameValueString to delete a retained publication
Public void deletePublication (String topic, String stream, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> topic - (String) Topic of publication to delete. stream - (String) Stream of publication to delete. msg - (MQMessage) The message object in which the name value string is stored. Throws Exception- - Any exceptions thrown by the underlying string or MQSeries message classes are re-thrown. 	This function builds NameValueString to delete a retained publication
public int deregisterPublisher(String topic, String stream, String qname, String qmname, byte[] buffer) throws Exception	
<ul style="list-style-type: none"> topic - (String) Topic to deregister. stream - (String) Stream to deregister. qname - (String) Name of queue where confirmations are to be sent. qmname - (String) Name of queue manager where confirmations are to be sent. Buffer - (byte[]) The buffer in which the name value string is stored. Returns int – The length of the publish/subscribe header. Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown. 	This function builds NameValueString to indicate that a publisher shall no longer publish data for the topic indicated
public void deregisterPublisher(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception	
<ul style="list-style-type: none"> topic - (String) Topic to deregister. stream - (String) Stream to deregister. qname - (String) Name of queue where confirmations are to be sent. 	This function builds NameValueString to indicate that a publisher shall no longer publish data for the topic indicated

Pub-Sub Helpers (Bindings for C Applications)	
<ul style="list-style-type: none"> • qmname - (String) Name of queue manager where confirmations are to be sent. • Msg - (MQMessage) The message object in which the <i>name value</i> string is stored. • Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown. 	
<pre>public int deregisterSubscriber(String topic, String stream, String qname, String qmname, byte[] buffer) throws Exception</pre>	
<ul style="list-style-type: none"> • topic - (String) Topic to deregister. • stream - (String) Stream to deregister. • qname - (String) Name of queue where publications are sent. • qmname - (String) Name of queue manager where publications are sent. • buffer - (byte[]) The buffer in which the name value string is stored. • Returns int - The length of the publish/subscribe header. • Throws Exception- Any exceptions thrown by the underlying string or MQSeries message classes are re-thrown. 	This function builds NameValueString to indicate that a subscriber shall no longer receive data for the topic indicated
<pre>public void deregisterSubscriber(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception</pre>	
<ul style="list-style-type: none"> • topic - (String) Topic to deregister. • stream - (String) Stream to deregister. • qname - (String) Name of queue where publications are sent. • qmname - (String) Name of queue manager where publications are sent. • msg - (MQMessage) The message object in which the <i>name value</i> string is stored. • Throws Exception- Any exceptions thrown by the underlying string classes are re-thrown. 	This function builds NameValueString to indicate that a subscriber shall no longer receive data for the topic indicated
<pre>public int publish(String topic, String qname, String qmname, String puboptions, byte[] buffer) throws Exception</pre>	
<ul style="list-style-type: none"> • topic - (String) the topic of the publication. A topic shall be provided. • qname - (String) the queue named on the registerPublisher call. • qmname - (String) the queue manager named on the registerPublisher call. • puboptions - (String) MQPS_RETAIN_PUBLICATION or none. • buffer - (byte[]) The buffer in which the <i>name value</i> string is stored. • Returns int - The length of the publish/subscribe header • Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown. 	This function builds NameValueString to publish a publication

Pub-Sub Helpers (Bindings for C Applications)	
<pre>public byte [] publishWithData(String topic, String qname, String qmname, String puboptions, String pubData) throws Exception</pre> <ul style="list-style-type: none"> topic - (String) the topic of the publication. A topic shall be provided. qname - (String) the queue named on the <i>registerPublisher</i> call. qmname - (String) the queue manager named on the <i>registerPublisher</i> call. Puboptions - (String) MQPS_RETAIN_PUBLICATION or none. pubData -(String) The message data to <i>publish</i>. Returns byte[] - The buffer in which the <i>name value</i> string is stored. Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown. 	This function builds NameValueString to publish a publication including the publication data
<pre>public void publish(String topic, String qname, String qmname, String puboptions, MQMessage msg) throws Exception</pre> <ul style="list-style-type: none"> topic - (String) the topic of the publication. A topic shall be provided. qname - (String) the queue named on the <i>registerPublisher</i> call. qmname - (String) the queue manager named on the <i>registerPublisher</i> call. puboptions - (String) MQPS_RETAIN_PUBLICATION or none. msg - (MQMessage) The message object in which the <i>name value</i> string is stored. Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown. 	This function builds NameValueString to publish a publication
<pre>public int registerSubscriber(String topic, String stream, String qname, String qmname, byte[] buffer) throws Exception</pre> <ul style="list-style-type: none"> topic - (String) The topic being <i>subscribed to</i>. stream – (String) the stream being <i>subscribed to</i>. qname - (String) the name of the queue where messages are to arrive. qmname - (String) the name of the queue manager where messages are to arrive. Buffer - (byte[]) The buffer in which the <i>name value</i> string is stored. Returns int – The length of the publish/subscribe header. Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown. 	This function builds NameValueString to indicate that a subscriber shall receive data for the topic indicated
<pre>public void registerSubscriber(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception</pre> <ul style="list-style-type: none"> topic - (String) The <i>subscribed</i> topic. stream - (String) the stream being <i>subscribed to</i>. qname - (String) the name of the queue where 	This function builds NameValueString to indicate that a subscriber shall receive data for the topic indicated

Pub-Sub Helpers (Bindings for C Applications)	
<p>messages are to arrive.</p> <ul style="list-style-type: none"> • qmname - (String) the name of the queue manager where messages are to arrive. • msg - (MQMessage) The message object in which the name value string is stored. • Throws Exception- Any exceptions thrown by the underlying string or MQSeries message classes are re-thrown. 	
<pre>public int requestUpdate(String topic, String stream, String qname, String qmname, byte[] buffer) throws Exception</pre>	<p>This function builds NameValueString to request retained publications for the topic indicated</p>
<ul style="list-style-type: none"> • topic - (String) The topic to receive updates. • Stream - (String) The stream to receive updates. • qname - (String) the name of the queue where the updates are to arrive. • qmname - (String) the name of the queue manager where the updates are to arrive. • Buffer - (byte[]) The buffer in which the <i>name value</i> string is stored. • Returns int – The length of the publish/subscribe header. • Throws Exception- Any exceptions thrown by the underlying <i>string</i> or MQSeries message classes are re-thrown. 	
<pre>public void requestUpdate(String topic, String stream, String qname, String qmname, MQMessage msg) throws Exception</pre>	<p>This function builds NameValueString to request retained publications for the topic indicated</p>
<ul style="list-style-type: none"> • topic - (String) The topic to get updates. • stream - (String)) The stream to get updates for. • qname - (String) the name of the queue where the updates are to arrive. • qmname - (String) the name of the queue manager where the updates are to arrive. • msg - (MQMessage) The message object in which the <i>name value</i> string is stored. • Throws Exception-A re-thrown exception thrown by the underlying string or MQSeries message classes. 	

2.5 com.lmfs.framework.servlet.IFServlet

In the **Servlet Structure** (hierarchy) shown in 2, the **PDCServlet** is the base class for the PDC Application. **IFServlet** is the Framework *Base* class which all IF Servlet Applications should extend. The purpose of the **IFServlet** is to allow common function (methods) and framework integration as well as security across all applications. It allows abstract methods to be implemented down the hierarchy. An abstract class is one that has to be implanted by the class, which extends from it.

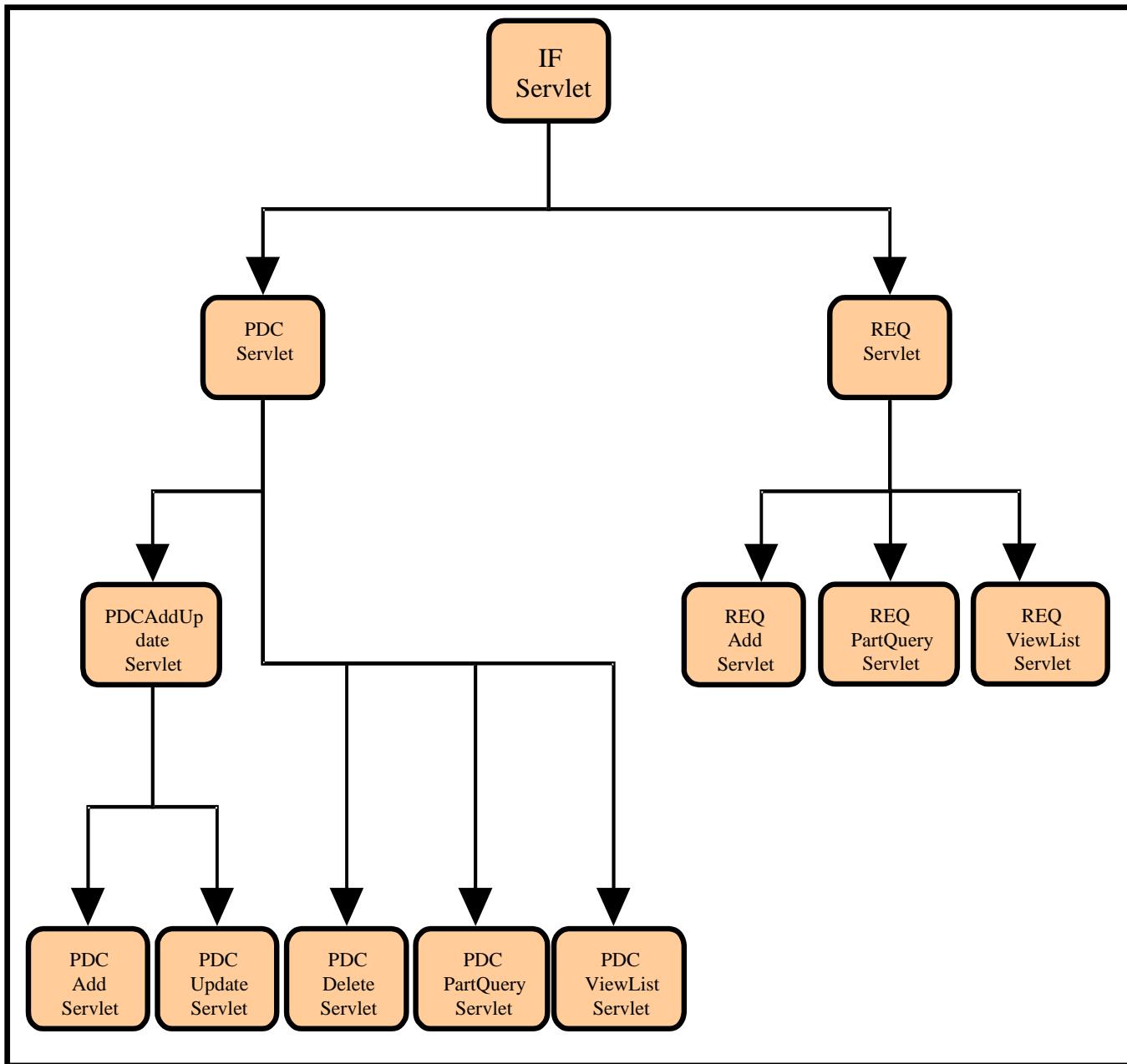


Figure 18 IFServlet Extended Class Diagram

Table 6: com.lmfs.framework.servlet.IFServlet

com.lmfs.framework.servlet.IFServlet	
public void doGet(HttpServletRequest req, HttpServletResponse res)	
<ul style="list-style-type: none"> Req - (HttpServletRequest) Object that encapsulates the request to the Servlet. Res - (HttpServletResponse) Object that encapsulates the response from the Servlet. 	Process incoming HTTP GET requests specific to all IFServlets
Public abstract void doGetService(HttpServletRequest request, HttpServletResponse response) throws Exception	
<ul style="list-style-type: none"> Request - (HttpServletRequest) Object that encapsulates the request to the Servlet. Response - (HttpServletResponse) Object that encapsulates the response from the Servlet. 	An abstract method, that Servlets derived from this base shall implement, since it is called in the above doGet method. Its purpose is to handle the incoming get requests specific to an application.
public void doPost(HttpServletRequest req, HttpServletResponse res)	
<ul style="list-style-type: none"> req - (HttpServletRequest) Object that encapsulates the request to the Servlet. res - (HttpServletResponse) Object that encapsulates the response from the Servlet. 	Process incoming HTTP POST requests
public abstract void doPostService(HttpServletRequest request, HttpServletResponse response) throws Exception	
<ul style="list-style-type: none"> request - (HttpServletRequest) Object that encapsulates the request to the Servlet. Response - (HttpServletResponse) Object that encapsulates the response from the Servlet. 	An abstract method, that Servlets derived from this base shall implement, since it is called in the above doPost method. Its purpose is to handle the incoming post requests specific to an application.
Protected HttpSession getHttpSession(HttpServletRequest request)	
<ul style="list-style-type: none"> Request - (HttpServletRequest) Object that encapsulates the request to the Servlet. Returns HttpSession - The session information extracted from the request. 	Creates a new session.
public java.lang.String getParameter(HttpServletRequest request, String parameterName, boolean checkRequestParameters, boolean checkInitParameters, boolean isParameterRequired, String defaultValue) throws Exception	
<ul style="list-style-type: none"> Request - (HttpServletRequest) Object that encapsulates the request to the Servlet. ParameterName - (String) The name of the parameter value to return. CheckRequestParameters - (boolean) When true the request parameters <i>are</i> searched . When false the request parameters <i>are not</i> searched. CheckInitParameters - (boolean) When true the Servlet init parameters <i>are</i> searched. When false init parameters <i>are not</i> searched. IsParameterRequired - (boolean) When true an exception is thrown when the parameter cannot be found. If false no exception shall be thrown if parameter is not found. DefaultValue - (String) The default value to return when the parameter is not found. Returns java.lang.String - The value of the specified parameter. 	Returns the value of the requested parameter identified by the parameter name specified.
protected SessionInfoStruct getSessionInfoStruct(HttpServletRequest request, String location) throws Exception	
<ul style="list-style-type: none"> request - (HttpServletRequest) Object that encapsulates the request to the Servlet. Location - based on the user making the request. 	Creates a security related SessionInfoStruct from EXPECTED parameters taken from the Servlet request. The EXPECTED Servlet request parameters are: LDAP

com.lmfs.framework.servlet.IFServlet	
<ul style="list-style-type: none"> • BaseNameQualifier – based on the user making the request • Returns SessionInfoStruct - The completed <i>SessionInfoStruct</i> that is required for method invocations to Component Broker. This is used to check access privileges of the user making the request. 	<p>Groups and PAC Credentials. The menu system or JSPs need to supply the "location" and BaseNameQualifier. The Framework currently retrieves this info from LDAP in the Menu URLs as parameters. A users menu is filtered by security checks as to what he has access to, this is how location and BaseNameQualifier is related to a specific user.</p>
public java.lang.Object getValueFromSession(HttpServletRequest request, String key)	
<ul style="list-style-type: none"> • request - (HttpServletRequest) Object that encapsulates the request to the Servlet • key - (String) The identifier of the value to be retrieved from the HTTP session. 	Retrieves an object, specified by the parameter <i>key</i> , from the session that is contained in the request.
public void init(ServletConfig config) throws javax.servlet.ServletException	
<ul style="list-style-type: none"> • config - (ServletConfig) The <i>GenericServlet</i> interface is passed to the super class. 	Contains the common code required to initialize the Servlets. The logging service is initialized. The connection to the Component Broker environment is established. If security between the Servlet engine and the Component Broker environment is turned on, the ServletLogonHelper is invoked to perform the login into Component Brokers' DCE cell. Typically, this method shall not be overridden by the client Servlet.
protected abstract void initializePropertyManager()	
<ul style="list-style-type: none"> • N/A 	An abstract method, Servlets derived from this base case shall provide the implementation. The purpose is to insure each application gets client properties based on unique file names. The IFFilePropertyManager class is used for getting properties (this class wraps the PropertyResourceBundle) which are used to configure the application at runtime.
protected abstract void initializeServerConnection()	
<ul style="list-style-type: none"> • N/A 	<p>An abstract method, Servlets derived from this base case shall provide the implementation. Its purpose is to allow the deriving class to implement a specific server connection policy.</p> <p>This abstract method is the same for both CORBA and EJB components that require the initialization of the CBSeriesGlobal Object for security.</p> <p>(i.e. CORBA & EJB: CBSeriesGlobal.Initialize(args, ORBProperties))</p>
public void putValueOnSession(HttpServletRequest request, String key, Object object)	
<ul style="list-style-type: none"> • request - (HttpServletRequest) Object that encapsulates the request to the Servlet. • key - (String) The value that is used to identify the object being place on the session object. • object - (Object) The object to be placed on the session. 	Places the given object on the HTTP session, contained in the request object, using the key specified as the name for future retrieval.

2.6 com.lmfs.framework.* - Servlet Utility Helpers

Table 7: com.lmfs.framework.* - Servlet Utility Helpers

com.lmfs.framework.* - Servlet Utility Helpers	
Abstract class IFPropertyManager	
public IFPropertyManager(String resourceName)	
• ResourceName	Abstract Wrapper class used to get resources It is not called directly by IF developers
public abstract String getString(String key)	
• Key - (String) bundle key	Wrappers bundle getString
• Returns String	
class IFFilePropertyManager	
public IFFilePropertyManager(String resourceName)	
• resourceName - (String) resource name	Wrapper class used to get bundle resources from a file
public String getString(String key)	
• key - (String) bundle key	Wrappers bundle getString
Returns String	
class ServletLoginHelper	
synchronized public boolean checkLogin(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, String userID, String password) throws LoginHelperGeneralException, LoginHelperBadCredentialsException, LoginHelperLoginFailedException, ServletException, IOException	
• HttpServletRequest - (HttpServletRequest) Object that encapsulates the request to the Servlet.	Checks the session that is contained in the Servlet request to see if the user has previously logged in and that the login is still valid. If the user has not been logged in before, or the session is no longer valid, the user shall be logged in using the <i>userID</i> and <i>password</i> provided
• httpResponse - (HttpServletResponse) Object that encapsulates the response from the Servlet.	
• userID - (String) The <i>userID</i> to be used to login to the Component Broker DCE cell.	
• Password - (String) The <i>password</i> to be used to login to the Component Broker DCE cell.	
• Throws LoginHelperGeneralException – Is thrown in <i>login helper</i> when errors are of general nature, passed to <i>invoking</i> class.	
• Throws LoginHelperBadCredentialsException - Is thrown in <i>login helper</i> when errors are Bad Credentials passed to <i>invoking</i> class.	
• Throws LoginHelperLoginFailedException – Is thrown in <i>login helper</i> when errors deal with Login failure , passed to <i>invoking</i> class.	
• Throws ServletException – Servlet related errors are passed to <i>invoking</i> class.	
• Throws IOException – IO errors are passed to <i>invoking</i> class	
Protected Object doLogin(String string1, String string2) throws LoginHelperLoginFailedException	
• string1 - (String) The <i>userID</i> to be used to login to the Component Broker DCE cell.	Logs the user into the Component Broker DCE cell using the <i>userID</i> and <i>password</i> provided.
• String2 - (String) The <i>Password</i> to be used to login to the Component Broker DCE cell.	
• Returns Object - The security level 2 credentials that result from a successful login.	
• Throws LoginHelperLoginFailedException – Is thrown in <i>login helper</i> when errors deal with	

com.lmfs.framework.* - Servlet Utility Helpers	
<i>Login failure</i> , passed to <i>invoking</i> class.	
public void init() throws LoginHelperGeneralException	
<ul style="list-style-type: none">Throws LoginHelperGeneralException – Is thrown in <i>login helper</i> when errors are of general nature, passed to <i>invoking</i> class.	Sets up the connection to the Component Broker security context and creates the <i>system login helper</i> that shall ultimately be used to log users into the Component Broker DCE cell.
public static void logoff(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException	
<ul style="list-style-type: none">HttpServletRequest - (HttpServletRequest) Object that encapsulates the request to the Servlet.HttpServletResponse - (HttpServletResponse) Object that encapsulates the response from the Servlet.Throws ServletException - Servlet related errors are passed to <i>invoking</i> class. Throws <i>IOException</i> – IO errors are passed to <i>invoking</i> class.	Logs the user out of the Component Broker DCE cell by removing the credentials from the session contained in the Servlet request.

3. Security Helpers

3.1 com.lmfs.framework.LDAPHelper.*

This package provides wrappers that assist in the interface to an *LDAP Directory Server*. The package was developed for the Menu test component to access an LDAP server. It was originally designed on and used in a Windows NT/x86 environment, and was also successfully tested on Solaris 2.7. The level of abstraction it provides greatly reduces the level of effort required when connecting to an LDAP server, but proportionally limits the flexibility available. It may or may not be used for other LDAP access.

The classes provided in this package are:

com.lmfs.framework.LDAPHelper.LDAPHelper

LDAPHelper provides abstracted access to an LDAP Server using a properties file.

com.lmfs.framework.LDAPHelper.LDAPResultsHelper

Translates an **LDAPSearchResults** object such that it is easier to sort/navigate.

com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator

A class used with the menu test component for sorting purposes.

com.lmfs.framework.LDAPHelper.LDAPTests

A class used for menu test component debugging purposes.

3.1.1 com.lmfs.framework.LDAPHelper.LDAPHelper

LDAPHelper is a class to facilitate connecting to and searching an *LDAP Server*. **LDAPHelper** uses a mixture of a properties file and various constructors. The properties file is read by default in the beginning of all constructors, and the constructor parameters override the values read from the properties file as described.

The example below shows initiating the LDAPHelper that obtains information about the LDAP directory it is connecting to from the GCSSRegistry.properties file. The starting point for our menu test component, and **request_string** in this example, is "**ifmit=Main Menu,ou=Integration Framework,ou=usaf,ou=pki,ou=dod,o=U.S. Government,c=us**". The menu system test component uses the ifmit object class for the menu items. Therefore the **pattern_string** is set to "**ifmit=***" in this example. The result is a set list of the distinguished names of all of the menu items found below our the starting point, that is the **request_string**, in the LDAP directory.

```
private Vector loadMenu() {
    test();
    LDAPHelper ldh = null;
    try {
        ldh = new LDAPHelper();
    }
    catch (java.lang.Exception e) {
        menuLogger.error("Error constructing LDAP Helper " + "object in loadmenu():\n");
        e.printStackTrace();
    }
    LDAPSearchResults aSearchResults = ldh.findDN(REQUEST_STRING, PATTERN_STRING);
```

Figure 19: LDAPHelper Example from Menu.java file of the Menu System Test Component

Table 8: com.lmfs.framework.LDAPHelper.LDAPHelper

com.lmfs.framework.LDAPHelper.LDAPHelper	
public LDAPHelper ()	Constructor uses <GCSSRegistry.properties> file for all settings.
public LDAPHelper (int)	<ul style="list-style-type: none">In inSearchScope - The search scope to search with. Overrides corresponding GCSSRegistry.properties setting Constructor uses <GCSSRegistry.properties> file for all but scope setting.
private LDAPHelper (int, java.lang.String)	<ul style="list-style-type: none">In ReferralFlagIn inStrReferralURL - The URL of the referred-to LDAP Server. This stub is a placeholder for the handling of referrals.
public LDAPHelper (java.lang.String)	<ul style="list-style-type: none">In iniFileLocation - A subdirectory in the classpath that contains the GCSSRegistry.properties file. Constructor uses <GCSSRegistry.properties> file for all but the properties file location. The properties file shall still be named <GCSSRegistry.properties> but can be

com.lmfs.framework.LDAPHelper.LDAPHelper	placed in a different subdirectory name. Example: LDAPHelper x = new LDAPHelper("/SubDirName") , where the properties file should be located in some directory in the classpath as: /SubDirName/GCSSRegistry.properties .
public LDAPHelper (java.lang.String, int)	<ul style="list-style-type: none"> • In inStrHost - The host to connect to. Overrides GCSSRegistry.properties setting. • In inIntPort - The port to connect to. Overrides GCSSRegistry.properties setting.
public LDAPHelper (java.lang.String, int, int)	<ul style="list-style-type: none"> • In inStrHost - The host to connect to. Overrides corresponding GCSSRegistry.properties setting. • In inIntPort - The port to connect to. Overrides corresponding GCSSRegistry.properties setting. • In inIntSearchScope - The search scope to search with. Overrides corresponding GCSSRegistry.properties setting.
Private check_result ()	This private method verifies that an LDAP operation succeeded, and logs a warning if not.
private closeConnection ()	Private function to close connection to LDAP host.
private createConnection ()	Private function to create connection to LDAP host.
public finalize ()	Finalizes private attribute ldapConn_ .
public findDN (java.lang.String, java.lang.String)	<ul style="list-style-type: none"> • In inStrSearchBase - Base DN at which to begin the search. • In inStrFilt - The filter to search with, in format "<i>attribute=value</i>".
public getHost ()	Returns the host attribute of LDAPConnection .
private readINI ()	Sets all default settings from <GCSSRegistry.properties>

3.1.2 com.lmfs.framework.LDAPHelper.LDAPResultsHelper

Translates a **LDAPSearchResults** object so that it is easier to sort and navigate. Methods **NextEntry** and **NextAttribute** read the next **LDAPEntry** and next attribute of the current entry from the returned **LDAPSearchResults**, respectively.

The following code example takes the list of distinguished names returned by the **com.lmfs.framework.LDAPHelper**. *LDAPHelper.findDN* method and reads all of the attributes and values from the LDAP Directory for each Distinguished Name in the list.

```
LDAPResultsHelper mh = new LDAPResultsHelper(aSearchResults);
Vector infoltems = new Vector();
int currentItemLevel = 0;
mh.nextEntry();
mh.nextAttribute();
while (mh.getDN() != null) {
    LDAPInfoItem itemInfo = new LDAPInfoItem();
    itemInfo.setDn(mh.getDN());
    while (mh.getAttributeName() != "") {
        itemInfo.addAttribute(mh.getAttributeName(), mh.getAttributeValue());
        mh.nextAttribute();
    }
    if (!itemInfo.getItemLabel().equals("Main Menu"))
        infoltems.addElement(itemInfo);
    mh.nextEntry();
    mh.nextAttribute();
}
return createMenu(authorizeMenuItems(infoltems));
```

Figure 20: LDAPResultsHelper Code Example from the Menu.java file of the Menu System Test Component

Table 9: com.lmfs.framework.LDAPHelper.LDAPResultsHelper

com.lmfs.framework.LDAPHelper.LDAPResultsHelper	
public LDAPResultsHelper (netscape.ldap.LDAPSearchResults)	
<ul style="list-style-type: none"> In netscape.ldap. LDAPSearchResults inLDAPRes - The <i>resultset</i> to enumerate. 	Constructs a new LDAPSearchResults object accepting an LDAPSearchResults object as a parameter.
public LDAPResultsHelper (netscape.ldap.LDAPSearchResults, netscape.ldap.LDAPEntryComparator)	
<ul style="list-style-type: none"> In netscape.ldap. LDAPSearchResults inLDAPRes - The <i>resultset</i> to enumerate. In netscape.ldap. LDAPEntryComparator inComparator - The <i>comparator</i> to use when sorting the resultset. 	Constructs a new LDAPSearchResults object accepting an LDAPSearchResults object and an LDAPEntryComparator object as parameters.
public getAttributeName ()	Returns the name of the currently enumerated attribute.
public getAttributeValue ()	Returns the value of the currently enumerated attribute.
public getDN ()	Returns the DN of the currently enumerated LDAP Entry.
public nextAttribute ()	Enumerates to next attribute in the attribute set of the current LDAP Entry.
public nextEntry ()	Enumerates to next LDAPEntry in the entries of the LDAPResultSet .

3.1.3 com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator

The **LDAPMenuEntryComparator** class is not currently used but was originally specifically developed for the Menu System Test Component.

The class is used in conjunction with **LDAPResultsHelper** to sort results obtained from a **FindDN** call. The class may be used as an example of a custom LDAP search results sorting routine. The constructors usually accept a string parameter **BaseDN** that indicates the portion of the DN that should be ignored when the returned entries are compared.

The comparison within **isGreater()** is done in the following manner:

- The **BaseDN** portion of each **LDAPEntry**'s DN is removed
- The remainder of the DNs are converted to a Policy Director namespace object name format (**reverse order, /-delimited**. Example "**cn=myName,ou=Organization1**" is converted to "**/Organization1/myName**")
- The strings are subsequently compared.
- If the first string is greater than the second, true is returned.

Table 10: com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator

com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator	
Public LDAPMenuEntryComparator (java.lang.String)	
• IN PDMMenuBaseDN – The Base DN to ignore when sorting search results by DN.	Use the PDFormattedDN to compare search results from an LDAP Search.
Public LDAPMenuEntryComparator (java.lang.String, boolean, java.lang.String)	<p>• IN PDMMenuBaseDN – The Base DN to ignore when sorting search results by DN.</p> <p>• IN useSortAttribute – Boolean to indicate that the sort attribute should be used. The value is expected to be true and is ignored. The parameter is meant to distinguish between constructors.</p> <p>• IN sortAttributeToUse – String value to indicate which attribute contains leaf node sorting values.</p> <p>Use the sort attribute (when available) instead of the PDFormattedDN to compare search results from an LDAP Search. If the sort attribute is nonexistent, the Policy Director-Formatted DN is used instead.</p> <p>The standard comparison ensures that the order of the distinguished name begins with the outermost container (c=us,o=u.s. government,...,cn=Order Part) versus beginning with the relative distinguished name (e.g. cn=Order Part,...,c=us)</p> <p>The sortAttributeToUse identifies that the comparison should use the value of this attribute in the object referenced by the Distinguished Names for the comparison rather than the Distinguished Name itself.</p>
Public LDAPMenuEntryComparator (boolean, java.lang.String)	
• IN useSortAttribute – Boolean to indicate that the sort attribute should be used. The value is expected to be true and is ignored. The parameter is meant to distinguish between constructors.	Use the sort attribute (when available) instead of the Policy Director-Formatted DN to determine which one's greater.
• IN sortAttributeToUse – String value to indicate which attribute contains leaf node	The standard comparison ensures that the order of the distinguished name begins with the outermost container (c=us,o=u.s. government,...,cn=Order Part) versus

com.lmfs.framework.LDAPHelper.LDAPMenuEntryComparator	
sorting values.	beginning with the relative distinguished name (e.g. cn=Order Part,...,c=us)
The sortAttributeToUse identifies that the comparison should use the value of this attribute in the object referenced by the Distinguished Names for the comparison rather than the Distinguished Name itself.	
Public isGreater (netscape.ldap.LDAPEntry, netscape.ldap.LDAPEntry)	<ul style="list-style-type: none"> • IN LDAPEntry - The first entry to compare. • IN LDAPEntry - The second entry to compare to the first. <p>Required method for an LDAPComparator. Returns true if the first entry is greater than the second. LDAPMenuEntryComparator first uses the sort attribute if the useSortAttribute flag is set, followed by a Policy Director formatting of the DN.</p>

3.1.4 com.lmfs.framework.LDAPHelper.LDAPTests

LDAPTests A class used for menu test component debugging purposes only.

LDAPTests lists all entries with the DN prefix of "ifmit=*" after the hardcoded base DN: "**ifmit=Main Menu,ou=Integration Framework, ou=usaf, ou=pki, ou=dod, o=U.S.Government, c=usa.**" **LDAPTests** is meant for debugging purposes.

Table 11: com.lmfs.framework.LDAPHelper.LDAPTests

com.lmfs.framework.LDAPHelper.LDAPTests	
public LDAPTests ()	
public static main (java.lang.String[])	
• IN initArgs[] - unused in this test routine.	main() is meant as a debugging tool. It uses LDAPHelper and LDAPResultsHelper to dump out everything with "ifmit=*" after hardcoded base DN: " ifmit=Main Menu,ou=Integration Framework, ou=usaf, ou=pki, ou=dod,o=U.S.Government, c=usa. "

3.2 com.lmfs.framework.security.*

This package provides classes that interface the authentication and access control portions of the Integration Framework. It was originally designed on and used in the WebSphere AE and EE products in a Windows NT/x86 environment. A stubbed version of the classes is available on the Solaris 2.7 platform for WebSphere AE and EE.

The level of abstraction it provides greatly reduces the level of effort required when interfacing with **aznAPI**, but proportionally limits the flexibility available.

This package also contains classes to export a structure from an LDAP server in Policy Director Object Name format.

The Security Package contains the following classes/packages:

com.lmfs.framework.security.IFSecurityException

An exception raised by **PDAuthz** subclasses.

com.lmfs.framework.security.IFMenuPOSCreator

A class used by the menu test components. It exports LDAP entries to a format readable by Policy Director.

com.lmfs.framework.security.PDCheckParser

A class that filters HTML documents by replacing HTML tags with Policy Director values. This class is untested.

com.lmfs.framework.security.PDCheckEnvironment

A class which contains data necessary for PDCheckParser.

com.lmfs.framework.security.AuthnInfo

Is an interface for retrieving authentication information.

3.2.1 com.lmfs.framework.security.IFSecurityException

An exception raised when a security error occurs. Stores additional exception information as needed.

Table 12: com.lmfs.framework.security.IFSecurityException

com.lmfs.framework.security.IFSecurityException	
public IFSecurityException ()	
public IFSecurityException (java.lang.String)	
• IN s - Additional exception information.	Constructor allowing additional information to be passed as a string parameter.

3.2.2 com.lmfs.framework.security.IFMenuPOSCreator

IFMenuPOSCreator exports the LDAP items used by the menu system to a file readable by Policy Director. Uses the **com.lmfs.framework.LDAPHelper.*** classes to accomplish LDAP communication. See the routine **com.lmfs.framework.util.StringExt.getPDFFormattedDN** (String, String) for information on how this is done.

This is a self contained executable created for the express purpose of generating a text file of the menu system hierarchy. The text file is then placed on the Policy Director Security Management Server so that the Menu can be viewed in the Policy Director Object Namespace and to allow Access Control Lists to be applied to it.

Table 13: com.lmfs.framework.security.IFMenuPOSCreator

com.lmfs.framework.security.IFMenuPOSCreator	
Public IFMenuPOSCreator ()	Default Constructor.
Public static main (java.lang.String[])	
• IN args – Command-line arguments.	Method called from command line invocation. Creates PD Object Namespace from a DN in LDAP.

3.2.3 com.lmfs.framework.security.PDCheckParser

PDCheckParser provides filtering of HTML documents. This class is not used by any framework or Test Component code.

The **PDCheckParser** replaces some basic HTML tags as can be seen in the following example:

<IVUSER> is replaced with the currently logged on user.
<IVGROUPS> is replaced with the currently logged on user's group list.
<IVPAC> is replaced with the user's stringified PAC.
In addition fine-grained authorization semantics are supported, delimited by <IVCHECK> and </IVCHECK> tags. <IVCHECK> provides "IF" semantics while </IVCHECK> provides "ENDIF" semantics. Also available is an optional <IVFAIL> tag, which provides "ELSE" semantics.

An <IVCHECK> tag specifies that any text between <IVCHECK> and </IVCHECK> is only available to users who pass the authorization checks listed in the <IVCHECK> tag.

For users who do not pass the checks, an optional <IVFAIL> tag is available. Text between the <IVFAIL> and </IVCHECK> tags are available to the users who do not pass the authorization check.

The format is:

```
<xmp>
<IVCHECK object="objname">
    Text that is only available to a user with access
<IVFAIL>
    Text that is only available to a user WITHOUT access
</IVCHECK>
</xmp>
```

where objname is a fully qualified protected element in the PD namespace.

If the user requesting the page has read access for that object then the text within the <IVCHECK> and </IVCHECK> tags shall be returned. Otherwise, it shall be skipped and the text between the <IVFAIL> and </IVCHECK> tags shall be returned. The <IVFAIL> tag is optional.

```
An example:  
<html><body>  
<h1>This heading can be seen by anyone</h1>  
This is a line of text that anyone can see.  
<IVCHECK object="/stuff/foo">  
Congratulations, you have permission to access /stuff/foo.  
<IVFAIL>  
Sorry, you do not have access to /stuff/foo.  
</IVCHECK>  
<P>This text can be seen by anyone  
</body></html>
```

Figure 21: Example of PDCheckParser Code

The following restrictions shall be obeyed, otherwise the **<IVCHECK>**, **<IVFAIL>** and **</IVCHECK>** tags shall not be handled correctly:

- The **<IVFAIL>** tag is optional.
- The tags shall be on a single line.
- The tags shall not span lines.
- The tags shall be the only things on the line.
- **<IVCHECK>** and **<IVFAIL>** may be in lower, upper, or mixed case.
- The keyword **<i>object</i>** on the **<IVCHECK>** tag may be in lower, upper, or mixed case.
- You should place the object name inside of double quotes.
- The **<IVCHECK>** tags may be nested. If you are inside of an **<IVCHECK>** tag that failed authorization, however, you cannot access data even if that data is inside of an **<IVCHECK>** tag that would pass (unless you are inside an **<IVFAIL>** tag).

Example:

```
<IVCHECK object="/data/that/I/cannot/access">  
I shall not see this line.  
<IVCHECK object="/data/that/I/can/access">  
Normally I would have access to this, but since I am inside of  
another tag that failed, I shall not see this.  
</IVCHECK>  
<IVFAIL>  
<IVCHECK object="/data/that/I/can/access">  
Since I am inside of an IVFAIL, I can see this.  
</IVCHECK>  
</IVCHECK>  
<IVCHECK object="/data/that/I/can/access">  
I see this line.  
</IVCHECK>
```

Figure 22: Example of **<IVCheck>** HTML Tag

Table 14: com.lmfs.framework.security.PDCheckParser

<code>com.lmfs.framework.security.PDCheckParser</code>	
<pre>public PDCheckParser (com.lmfs.framework.security.PDAuthz.PDCheckAuthz, java.lang.String, java.lang.String, java.lang.String, java.lang.String)</pre>	
<ul style="list-style-type: none"> • In templateFilename <ul style="list-style-type: none"> - The template file name 	Constructor that records the filename of the template file.
<code>private filterHTML (java.io.BufferedReader, java.io.BufferedWriter)</code> <ul style="list-style-type: none"> • IN r - The reader from which we read the HTML text • IN w - The writer to which we write the filtered HTML text 	<p>The filterHTML command shall write the processed HTML to the output stream. It shall search for tags that are completely on a single line (that is, the "<" and ">" are both on the same line).</p> <p>*If the tag is one that we care about, then we handle it. Otherwise, we try to write it to the output stream. Note that we may be inhibiting output, so it may not actually be written out if we are inside of a <IVCHECK> and </IVCHECK> pair that failed authorization.</p>
<code>private handleTag (java.lang.String, java.io.BufferedWriter)</code> <ul style="list-style-type: none"> • In line - A line of HTML data 	<p>The handleTag command shall scan a line for any of our supported tags and process it. If one of those tags is found, then handleTag shall return true to tell its caller that we handled the tag. When an <IVCHECK> tag is found, the following occurs:</p> <ul style="list-style-type: none"> • A table of arguments is built (keyword/value pairs) • An authorization check is done based on the arguments. • The inhibitOutput flag is set based on the result of the authorization check <p>With the <IVCHECK> there are two parameter combinations:</p> <ul style="list-style-type: none"> • Only objname is provided • The objname is not provided <p>When an <IVFAIL> tag is found the inhibitOutput flag is toggled.</p> <p>When an </IVCHECK> tag is found the inhibitOutput flag is set to its previous state</p> <p>When an <IVUSER> tag is found the current username is substituted for the tag</p> <p>When an <IVGROUPS> tag is found the current user's group list is substituted for the tag</p> <ul style="list-style-type: none"> • When an <IVPAC> tag is found the user's PAC is substituted for the tag
<code>private parseArgString (java.lang.String)</code> <ul style="list-style-type: none"> • In args - A line of name/value pairs, separated by equals signs 	<p>The command parseArgString shall scan a line for name/value pairs (separated by equals signs, “ = ”) and return a hashtable of those pairs.</p> <p>The value data may be enclosed in single or double</p>

com.lmfs.framework.security.PDCheckParser	quotes. In addition, the name (key) shall be uppercased before putting it into the table, to make later retrieval easier.
private setupStreamTokenizer (java.lang.String)	<ul style="list-style-type: none"> In args - A line of name and value pairs <p>SetupStreamTokenizer shall initialize a StreamTokenizer object for parsing <IVCHECK> argument strings. The basic form of an argument string is name=value where value may be in single or double quotes (and may contain spaces). There may be several name=value pairs in the string, each delimited by white space.</p>
public toStream (java.io.Writer)	<ul style="list-style-type: none"> In writer - The writer to which we shall send the processed HTML text <p>ToStream is used to write the HTML data represented by this component to the specified output writer. This class acts as a filter, allowing our super class to write its HTML to a writer that we provide. It then scans the text for tags, processing them, and writing the resulting HTML to the writer provided by our caller.</p>
private writeOutput (java.io.Writer, java.lang.String)	<ul style="list-style-type: none"> In w - The writer to which output is directed In s - A string to write <p>WriteOutput shall write a string to the writer if we are not inhibiting output. To make the source easier to read from the browser's view source option, a new line shall be appended to the line of text.</p>

3.2.4 com.lmfs.framework.security.PDCheckEnvironment

PDCheckEnvironment contains the data necessary to make authorization decisions through <IVCHECK> tags in an HTML document. It contains the notion of a current environment, and a stack of saved environments (to support nesting <IVCHECK> tags).

This class is used by the **PDCheckParser** class. It is not called by any Integration Framework or Test Components.

The environment contains the following data:

- Name of the object that we want to check access to
- Current state of inhibit output - i.e. are we between an <IVCHECK> and </IVCHECK> tag and if we are, did the authorization succeed or fail?
- A flag to indicate if we are currently within an <IVFAIL> block

Table 15: com.lmfs.framework.security.PDCheckEnvironment

com.lmfs.framework.security.PDCheckEnvironment	
public PDCheckEnvironment (com.lmfs.framework.security.PDAuthz.PDCheckAuthz, java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> In authChecker - The authorization interface. In user - The username of the user we are 	Constructor for PDCheckEnvironment . It shall save the application component (from which we get the security context), and create a new environment stack.

com.lmfs.framework.security.PDCheckEnvironment	
check authorization for.	
<ul style="list-style-type: none"> • In userpac - The PAC of the user we are check authorization for. This can be null, in which case we shall use the username for making auth API calls. 	
private checkAuth (java.lang.String, com.lmfs.framework.security.PDCheckEnvironment.StackElement)	<p>CheckAuth shall check authorization given an object name, and set the inhibitOutput indicator. Authorization is performed using the AuthAPI. The rules are as follows:</p> <ol style="list-style-type: none"> 1. If inhibitOutput is false (meaning that we are currently in data that we can see) then do auth check and set inhibitOutput based on that check. 2. If inhibitOutput is true (meaning that we are currently in data that we cannot see) then do not bother doing the auth check, or resetting inhibitOutput, since we shall inhibit output even if the auth check would have passed
public enterFailureSection ()	
	The enterFailureSection is provided to support the <IVFAIL> tag. It changes the current value of inhibitOutput , but does not affect any stacked values. It also remembers that we have seen an <IVFAIL> tag, so that we do not allow the user to have more than one <IVFAIL> in a single <IVCHECK> .
public getInhibitOutputState ()	
	GetInhibitOutputState is an accessor function to allow the caller to determine what the current environment's inhibit output state is.
public newEnv (java.lang.String)	
<ul style="list-style-type: none"> • In newObjName - the object name 	The newEnv tag shall save the current environment on the stack, and create a new current environment using the parameters passed.
private newObjectEnv (java.lang.String, com.lmfs.framework.security.PDCheckEnvironment.StackElement)	
<ul style="list-style-type: none"> • In newObjName - the object name • In e - the current stack element 	The newObjectEnv tag shall create a new current environment using the parameters passed. It shall determine if output is to be inhibited based on the <i>Check Authorization</i> on the objName (checkAuth shall determine whether to inhibit output).
public restoreEnv ()	
	The restoreEnv tag shall retrieve the last environment saved on the stack and make it the current environment.

3.2.5 com.lmfs.framework.security.AuthnInfo

AuthnInfo provides an interface for retrieving authentication information. This allows the MA to abstract the implementation of the actual authentication technique and shield developers from future API and Java wrapper changes.

Table 16: com.lmfs.framework.security.AuthnInfo

com.lmfs.framework.security.AuthnInfo	
public getClientIdent ()	
	Retrieves the <i>Policy Director ID</i> for the user. The returned PAC should be in a format ready for authorization checks.
public getGroups ()	Returns the Groups to which a given user belongs.
public hashCode ()	Returns a unique integer value derived from the Client Identity, Groups, and PAC. It has not been implemented.
public getPAC ()	Retrieves the <i>Policy Director PAC</i> for the user. The returned PAC should be in a format ready for call to com.ibm.pd.Authzn.Azn.pac_get_creds
public setClientIdent (java.lang.String)	<ul style="list-style-type: none"> In string - A string used to set the <i>Client Identity</i>. <p><i>Setter</i> function for the ClientIdent attribute.</p>
public setGroups (java.lang.String)	<ul style="list-style-type: none"> In java.lang.String - A string to set the Groups to. <p><i>Setter</i> function for the Groups attribute.</p>
public setPAC (java.lang.String)	<ul style="list-style-type: none"> In newPac – New PAC to set in AuthnInfo structure. <p><i>Setter</i> function for the PAC attribute.</p>

3.3 com.lmfs.framework.security.PDAuthn.*

This package provides classes that interface the authentication portions of the Integration Framework. It is used for authentication in the WebSphere AE environment. It should not be used in the WebSphere EE (CB) environment. Refer to Section 3.6 and Developer's Guide Section 5.3 for information on authenticating in the WebSphere EE (CB) environment.

The level of abstraction it provides greatly reduces the level of effort required when interfacing with the **aznAPI**, but proportionally limits the flexibility available.

The Security Package contains the following classes/packages:

com.lmfs.framework.security.PDAuthn.PDAuthnInfo

Interface for retrieving authentication information. The developer should call the methods on this interface or its parent interface, com.lmfs.framework.security.AuthnInfo.

com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl

A corresponding Policy Director implementation of the **AuthnInfo** interface.

com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl

A Servlet-centered implementation of the **PDAuthnInfo** interface

3.3.1 com.lmfs.framework.security.PDAuthn.PDAuthnInfo

PDAuthnInfo provides an interface for retrieving authentication information. This allows the MA to abstract the implementation of the actual authentication technique and shield developers from future API and Java wrapper changes.

PDAuthnInfo extends the **com.lmfs.framework.security.AuthnInfo interface**, adding the **LDAPAuthenticateUser()** method and defining a separate, extendible interface for Policy Director (PD) Authentication Routines.

Table 17: com.lmfs.framework.security.PDAuthn.PDAuthnInfo

com.lmfs.framework.security.PDAuthn.PDAuthnInfo	
public equals (java.lang.Object)	
• In pdAuthnInfoObject - an object also of type PDAuthnInfo to compare to the parent object of the method.	The equals method compares the attributes of the method's parent object to those of the object passed in the parameter, and returns true if all attribute values are the same.
Public getClientIdent ()	Returns the Client Identity.
Public getGroups ()	Returns the client Groups.
Public hashCode ()	Returns a hash of the Client Identity, PAC, and Groups.
Public getPAC ()	Returns the client's PAC.
Public LDAPAuthenticateUser (java.lang.String, java.lang.String)	
• In string - user DN to authenticate with. • In string - user <i>password</i> to authenticate with.	Provides a method of authentication using LDAP.
public setClientIdent (java.lang.String)	Sets the client Identity attribute.
public setGroups (java.lang.String)	Sets the client Groups attribute.
public setPAC (java.lang.String)	Sets the client PAC attribute.

3.3.2 com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl

PDAuthnInfoImpl provides an implementation of the **PDAuthnInfo** interface. It is used to store authentication information and to provide an authentication interface. It makes calls to, and is thus dependent on, **IV.Auth.AZNAPI (aznapi.class)**.

Refer to the following Section 3.3.3

com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl for code examples as

the **com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl** is extended by the **com.lmfs.framework.security.PDAuthn.Servlet.PdAuthnInfoImpl** class.

Table 18: com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl

com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl	
public PDAuthnInfoImpl ()	Default Constructor.
public equals (java.lang.Object)	<ul style="list-style-type: none"> In anObject – Object to compare to this object. Classname shall be the same. Returns true if the passed object's attributes are value-equivalent.
public init (com.lmfs.framework.util.PropertiesExt)	<ul style="list-style-type: none"> In com.lmfs.framework.util.PropertiesExt prp- PropertiesExt containing the properties to initialize the aznAPI with. Initialize the aznAPI using the propertiesExt passed.
public init (IV.Auth.AZNAPI)	<ul style="list-style-type: none"> In apiObj-A valid aznAPI to set to the aznAPI. Calls initApi to initialize the aznAPI (sets the azn object) using an already-initialized azn object). We assume that the sent aznAPI object is initialized.
public init (java.lang.String)	<ul style="list-style-type: none"> S java.lang.String-Name of aznAPI properties file. Initialize the aznAPI using the aznAPI properties file specified.
public init (java.util.Properties)	<ul style="list-style-type: none"> Prp java.util.Properties-Properties to initialize the aznAPI with. Initialize the aznAPI using the properties passed.
public init (java.util.ResourceBundle)	<ul style="list-style-type: none"> In java.util.ResourceBundle rb- Resource Bundle containing the properties to initialize the aznAPI with. Initialize the aznAPI using the ResourceBundle passed.
private initApi (com.lmfs.framework.util.PropertiesExt, IV.Auth.AZNAPI)	<ul style="list-style-type: none"> APIObject IV.Auth.AZNAPI Initialized aznAPI (sets the azn object) using an already-initialized azn object). We assume that the sent aznAPI object is already initialized if the properties parameter (1) is null.
public LDAPAuthenticateUser (java.lang.String, java.lang.String)	<ul style="list-style-type: none"> UserDN java.lang.String-DN of user to <i>Authenticate</i>. Password java.lang.String-Password of user to <i>Authenticate</i>. Provides a method of authenticating using LDAP.
public getClientIdent ()	Retrieves the Policy Director ID for the user. The returned PAC should be in a format ready for Authorization Checks.
public getGroups ()	Returns the groups to which a given user belongs.
public hashCode ()	Returns a unique integer value derived from the Client Identity, Groups, and PAC. Not Yet Implemented.
public getPAC ()	

com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl	
	Retrieves the Policy Director PAC for the user. The returned PAC should be in a format ready for call to com.ibm.pd.Authzn.Azn.pac_get_creds
<code>public setClientIdent (java.lang.String)</code>	
<ul style="list-style-type: none"> • In string - A string to set the <i>Client Identity</i> to. 	<i>Setter</i> function for the ClientIdent attribute.
<code>public setGroups (java.lang.String)</code>	
<ul style="list-style-type: none"> • In java.lang.String - A string to set the Groups to. 	Groups function for the Groups attribute.
<code>public setPAC (java.lang.String)</code>	
<ul style="list-style-type: none"> • In newPac – New PAC to set in AuthnInfo structure. 	Groups function for the PAC attribute.

3.3.3 com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl

PDServletAuthnInfoImpl is a Servlet implementation of the **PDAuthn** interface. It extends the com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl implementation of the com.lmfs.framework.security.PDAuthn abstract interface.

The following code example extracts the Policy Director username and credential from the HTTP header. The *getClientIdent* and *getPAC* methods are invoked by the Menu System and the IFServlet. The actual extensions created by this class are not used by the Integration Framework or the Test Components.

```
if (sessionInfoStruct == null)
{
    PDServletAuthnInfoImpl helper = new PDServletAuthnInfoImpl(request);
    String[] dsa1 = {" "}; //Dynamic String Array
    String[] dsa2 = {" "}; //Dynamic String Array
    String[] dsa3 = {" "}; //Dynamic String Array
    String[] dsa4 = {" "}; //Dynamic String Array

    String frameworkBaseNameQualifier = getParameter(request, "basenamequalifier", true,
true, true, null);
    //String frameworkBaseNameQualifier =
propertyManager.getString("frameworkBaseNameQualifier");
    UserSeclInfoStruct usec = new UserSeclInfoStruct(helper.getClientIdent(),
helper.getGroups(), helper.getPAC(), dsa1);
    UserSessionInfoStruct usess = new UserSessionInfoStruct(usec, dsa2);
    AppSessionInfoStruct asis = new AppSessionInfoStruct(location,
frameworkBaseNameQualifier, dsa3);
    sessionInfoStruct = new SessionInfoStruct(usess, asis, dsa4);
```

Figure 23: Examples of *getClientIdent*, *getGroups*, and *getPAC* from the IFServlet *getSessionInfoStruct* method

Table 19: com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl

com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl	
public PDServletAuthnInfoImpl (javax.servlet.http.HttpServletRequest)	
	Create a new User object from an HTTP request. Read all data from the HTTP header.
private extractUserGroupsFromRequest (javax.servlet.http.HttpServletRequest)	
• In req – The incoming HTTP request.	Retrieves the Policy Director username from the HTTP request.
private extractUserIdentFromRequest (javax.servlet.http.HttpServletRequest)	
• In req – The incoming HTTP request.	Retrieves the Policy Director username from the HTTP request.
private extractUserPACFromRequest (javax.servlet.http.HttpServletRequest)	
• In req - The incoming HTTP request.	Retrieves the Policy Director PAC for the user from the HTTP request, without the version information. The returned PAC should be in a format ready for call to azn_pac_get_creds.

3.3.4 com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl

PDAppAuthnInfoImpl is an application implementation of the **PDAuthn** interface. It extends the **com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl** implementation of the **com.lmfs.framework.security.PDAuthn** abstract interface.

Neither the Integration Framework nor its Test Components currently use this class. It should not be used at this time.

Table 20: com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl

com.lmfs.framework.security.PDAuthn.Application.PDAppAuthnInfoImpl	
public PDAppAuthnInfoImpl ()	Default Constructor..

3.3.5 com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl

PDEJBAuthnInfoImpl is an EJB implementation of the **PDAuthn** interface. It extends the **com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl** implementation of the **com.lmfs.framework.security.PDAuthn** abstract interface.

Neither the Integration Framework nor its Test Components currently use this class. It should not be used at this time.

Table 21: com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl

com.lmfs.framework.security.PDAuthn.EJB.PDEJBAuthnInfoImpl
public PDEJBAuthnInfoImpl ()
Default Constructor..

3.4 com.lmfs.framework.security.PDAuthz.*

This package provides classes that interface the authorization portions of the Integration Framework. It is used for authorization in the WebSphere AE environment. It should not be used in the WebSphere EE (CB) environment. Refer to appendix section 3.6 and Developer's Guide Section 5.7 for information on authenticating in the WebSphere EE (CB) environment.

The level of abstraction it provides greatly reduces the level of effort required when interfacing with the **aznAPI**, but proportionally limits the flexibility available.

The Security Package contains the following classes/packages:

com.lmfs.framework.security.PDAuthz.PDCheckAuthz

provides an interface for authorization checks. IF developers should use this interface as much as possible over specific implementations (below).

com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException

An exception only thrown when an authorization check specifically denies access to a resource for a user.

com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl

a generic implementation of the **PDCheckAuthz** interface for authorization checks. IF developers should use the **PDCheckAuthz** interface and this implementation over more specific implementations (below).

com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl

designates a package and class for authorization checks against *Component Broker* objects and classes.

com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl

designates a package and class for authorization checks against *Enterprise Java Bean (EJB)* components

com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl

a Servlet implementation of the **PDCheckAuthz** interface for Authorization checks

3.4.1 com.lmfs.framework.security.PDAuthz.PDCheckAuthz

PDCheckAuthz provides an interface for authorization checks. This allows the MA to abstract the implementation of the actual authorization engine to allow for ease of use within an application or Servlet. The application developer shall be padded from alterations to the **aznAPI** in future *Policy Director* release (such as when the *Native Interface* is removed.)

Table 22: com.lmfs.framework.security.PDAuthz.PDCheckAuthz

com.lmfs.framework.security.PDAuthz.PDCheckAuthz	
Public getAuthorizedObjects (java.lang.String, java.lang.String[], java.lang.String)	
<ul style="list-style-type: none"> • In infoImpl – Uses the creds stored in ai to accomplish the authorization check. • In objects – Checks each element in the array of string. • In permissions - Permissions to be checked on each object. 	Steps through a list of object names, determining the access decision for each, and returns the object names the user has access to in a new string array.
Public init (java.lang.String)	
<ul style="list-style-type: none"> • In initargs - A string to initialize from. 	Initializes any necessary class members using initargs . One necessary argument is AZNAPIPropertiesFileLocation , which should be passed as follows: AZNAPIPropertiesFileLocation=AZNAPI , or AZNAPI if it is the only argument.
public PDCheckAuthorization (java.lang.String, java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> • In username – The username of the individual for the authorization check. • In object - The object in the protected object namespace. • In Permset – The permissions being requested. 	PDCheckAuthorization performs an authorization check given a string username, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.
public PDCheckAuthorizationPAC (java.lang.String, java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> • In userpac – The PAC of the individual for the authorization check. • In object - The object in the protected object namespace. • In Permset – The permissions being requested. 	PDCheckAuthorization performs an authorization check given a string PAC for a user, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.

3.4.2 com.lmfs.framework.security.PDAuthz.PDAuthorizedException

PDAuthorizedException is an exception class for the **PDAuthz** interface methods that perform authorization checks. It shall only be thrown when an authorization check specifically denies access to a resource for a user. Errors are thrown via normal Exception objects.

Table 23: com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException

<code>com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException</code>	
<code>public PDAuthzUnauthorizedException (java.lang.String, java.lang.String, java.lang.String)</code>	
<ul style="list-style-type: none"> • In id – Client identity for which the authorization was checked. • In object - Object the client was attempting to access. • In Permset - the permissions the client desired on the object. 	The constructor takes as the parameters the information for the authorization check that resulted in the access denied. This could be used in an application-level audit trail.
<code>public PDAuthzUnauthorizedException (java.lang.String, java.lang.String, java.lang.String, java.lang.String[])</code>	
<ul style="list-style-type: none"> • In id – Client identity for which the authorization was performed. • In object - Object the client was attempting to access. • In Permset - permissions desired on the object. • In objects[] – String array of objects checked. 	The constructor takes as the parameters the information for the authorization check that resulted in the access denied. This could be used in an application-level audit trail.
<code>Public static formatStringArray (java.lang.String[])</code>	
<ul style="list-style-type: none"> • In objects - String array of objects. Returns a string of each element prepended with "Object: " and appended with "\n". 	The constructor takes as the parameters the information for the authorization check that resulted in the access denied. This could be used in an application-level audit trail.
<code>public toString ()</code>	
	Returns the exception as a string. The string representation is set during construction.

3.4.3 com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl

PDCheckAuthzImpl is an implementation of the **PDCheckAuthz** interface for authorization checks.

Usage:

1. Construct a **PDCheckAuthzImpl** object.
2. Call **PDCheckAuthzImpl.init(String)** with the NAME of the properties file that the implementation should use for **aznAPI** calls. The NAME refers to only the root of the filename. i.e. if the full file path is <c:\h\ifsservices\lib\AZNAPI_en_US.properties> leave off the entire path, locale information, and file information and send simply "aznAPI". The implementation shall search the classpath for a file with the specified root. (Example: **PDCheckAuthzImplObj.init("aznAPI")**);
3. Call 1 or more of **PDCheckAuthorization**, **PDCheckAuthorizationPAC**, or **getAuthorizedObjects** as needed.

This implementation is extended by
com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl. See section 3.4.6
com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl for code examples.

Table 24: com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl

com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl	
public PDCheckAuthzImpl ()	Default Constructor.
public getAuthorizedObjects (com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl, java.lang.String[], java.lang.String)	<ul style="list-style-type: none"> In infoImpl – Uses the creds stored in ai to accomplish the authorization check. In objects – Checks each element in the array of string. In permissions - Permissions to be checked on each object. Passes a list of objects to aznAPI.getAuthorizedObjects , which steps through the list of object names, determining the access decision for each, and returns the subset of object names the user has access to in a new string array.
public getAuthorizedObjects (java.lang.String, java.lang.String[], java.lang.String)	<ul style="list-style-type: none"> In pac - Uses the creds stored in pac to accomplish the authorization check. In objects – Checks each element in array objects. In permissions - Checks for all permissions on each object. Passes list of objects to AZNAPI.getAuthorizedObjects , which steps through the list of object names, determining the access decision for each, and returns the object names the user has access to in a new String array.
public init (java.lang.String)	<ul style="list-style-type: none"> In initargs - A string to initialize from. Initializes any necessary class members using initargs. One necessary argument is aznAPIPropertiesFileLocation , which should be passed as follows: AZNAPIPropertiesFileLocation= AZNAPI , or aznAPI if it is the only argument.
public isInitialized ()	Initializes any necessary class members using initargs. One necessary argument is aznAPIPropertiesFileLocation , which should be passed as follows: aznAPIPropertiesFileLocation=aznAPI resource which exists in the CLASSPATH Example: aznAPIPropertiesFileLocation=/aznAPI
public PDCheckAuthorization (com.lmfs.framework.security.PDAuthn.PDAuthnInfoImpl, java.lang.String, java.lang.String)	<ul style="list-style-type: none"> In authenObj – a PDAuthnInfoImpl object containing the valid credentials of the user for which the check shall be done In object - The object in the protected object namespace. In Permset - the permissions being requested on the object specified. <p>PDCheckAuthorization performs an authorization check given a string username, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.</p>
public PDCheckAuthorization (java.lang.String, java.lang.String, java.lang.String)	<ul style="list-style-type: none"> In username – The username of the individual for the authorization check. In object - The object in the protected object namespace. In Permset – The permissions being <p>PDCheckAuthorization performs an authorization check given a string username, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.</p>

com.lmfs.framework.security.PDAuthz.PDCheckAuthzImpl	
requested.	
public PDCheckAuthorizationPAC (java.lang.String, java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> • In userpac – The PAC of the individual for the authorization check. • In object - The object in the protected object namespace. • In Permset – The permissions being requested. 	
public setIsInitialized (boolean)	PDCheckAuthorizationPAC performs an authorization check given a PAC for a user's credentials, the object in the namespace and the permissions as a string. The method raises nothing on success. An exception is thrown if access is denied, or there is an error.
• In bool – Boolean value to set the isInitialized flag to.	Sets the isInitialized attribute (which is false by default) during initialization.

3.4.4 com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl

PDCBAuthzImpl designates a package and class for authorization checks against Component Broker objects and classes. It builds upon (extends) its parent package and implementation, **PDAuthz.PDCheckAuthzImpl**.

Neither the integration framework nor the test components currently implement this class. EJBs and CORBA Components should use the **IFCBSecurityInfo** Service for making authorization calls within EJB and CORBA Components.

Table 25: com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl

com.lmfs.framework.security.PDAuthz.CB.PDCBAuthzImpl	
Public PDCBAuthzImpl ()	Default Constructor.
Public init (java.lang.String)	<ul style="list-style-type: none"> • In initargs - A string to initialize from. <p>Initializes any necessary class members using initargs. One necessary argument is aznAPIPropertiesFileLocation, which should be passed as follows:</p> <p>aznAPIPropertiesFileLocation=RelativeLocationofPropertiesFileWithinClasspath</p> <p>Example:</p> <p>aznAPIPropertiesFileLocation=PropertyFiles\aznAPI</p>

3.4.5 com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl

PDEJBAuthzImpl designates a package and class for authorization checks against *Enterprise Java Bean (EJB)* components. It builds upon (extends) its parent package and implementation, **PDAuthz.PDCheckAuthzImpl**.

Neither the integration framework nor the test components currently implement this class. EJBs and CORBA Components should use the **IFCBSecurityInfo** Service for making authorization and authentication calls within EJB and CORBA Components.

Table 26: com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl

com.lmfs.framework.security.PDAuthz.EJB.PDEJBAuthzImpl	
public PDEJBAuthzImpl ()	Default Constructor.
public init (java.lang.String) <ul style="list-style-type: none">• In String initargs-A string to initialize from.	Initializes any necessary class members using initargs . One necessary argument is aznAPIPropertiesFileLocation , which should be passed as follows: aznAPIPropertiesFileLocation=CompleteFilePath Example: aznNAPIPropertiesFileLocation=C:\h\IFSServices\lib\aznAPI

3.4.6 com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl

PDServletAuthzImpl is a Servlet implementation of the **PDCheckAuthz** interface for authorization checks.

Usage:

1. Construct a **PDServletAuthzImpl** object.
2. Call **PDServletAuthzImpl.init(String)** with the NAME of the properties file that the implementation should use for **aznNAPI** calls. The NAME refers to only the root of the filename. i.e. if the full file path is **c:\h\ifsservices\lib\aznAPI_en_US.properties**, leave off the entire path, locale information, and file information and send simply **aznAPI**. The implementation shall search the classpath for a file with the specified root.
3. Call one or more of **PDCheckAuthorization**, **PDCheckAuthorizationPAC**, or **getAuthorizedObjects** as needed.

The following code example shows the construction of the **PDServletAuthzImpl** object and the initialization of the object.

```
public void init(javax.servlet.ServletConfig param1) throws javax.servlet.ServletException {  
    super.init(param1);  
    • • •  
    authObject = new com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl();  
    try {  
        menuServletLogger.info("Initializing the aznAPI...");  
    }
```

```
authObject.init("AZNAPIPropertiesFileLocation=AZNAPI");
} catch (java.lang.Exception e) {
    menuServletLogger.error("An error occurred initializing the aznAPI in MenuServlet.init()");
•••
}
•••
private Menu getUserMenu(HttpServletRequest request) {
•••
/*
If the session is new one or the user was changed:
    1. put the new user info value into the session
    2. create the new menu object based on the current user info
    3. put the created menu value into the session

If the session is not new one get the menu object from the session
*/
if (session.isNew()) {

    menuServletLogger.info("Session is NEW \n  Creating User Menu...");
    session.putValue(MenuServlet.SESSION_USER, aRequestUser);
    menu = new Menu(request, session, aRequestUser, authObject);
    session.putValue(MenuServlet.SESSION_MENU, menu);
} else {
    menuServletLogger.info("Recycling User Menu...");
    menu = (Menu) session.getValue(MenuServlet.SESSION_MENU);
}
return menu;
}
```

Figure 24: Example of Constructing and Initializing a PDServiceAuthzImpl Object taken from the MenuServlet.java File of the Menu System Test Component

The following code segment from the Menu System Test Component exercises the **PDCheckAuthorizationPAC** method and the Unauthorized Exception. None of the test components currently use the **PDCheckAuthorization** or **getAuthorizedObjects** methods.

```
public Menu( HttpServletRequest aRequest, HttpSession aSession,
            com.lmfs.framework.security.PDAuthn.Servlet.PDServiceAuthnInfoImpl aUser,
            com.lmfs.framework.security.PDAuthz.Servlet.PDServiceAuthzImpl inAuth ){

    if (menuLogger.isInfoEnabled()) menuLogger.info("Constructing menu for user " + aUser);
    auth = inAuth;
    CurrentUserSecInfo = aUser;
    menuItems = loadMenu();
•••

private class LDAPMenuLoader {
•••

private Vector authorizeMenuItems(Vector infoItems) {
•••

while (enum.hasMoreElements()) {
```

```

LDAPInfoItem item = (LDAPInfoItem) enum.nextElement();
if (menuLogger.isInfoEnabled()) menuLogger.info("Getting PD Formatted DN");
java.lang.String formattedDN = com.lmfs.framework.util.StringExt.getPDFormattedDN(item.dn,
pdMenuBase);
try {
    if (false) throw new
com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException("", "", "");
        if (menuLogger.isInfoEnabled()) menuLogger.info("About to check authorization, Creds=-
" + creds + "\n" +
        "                PD Name: " + "/IF Menu" + formattedDN + "\n" +
        "                Permission: " + "r");
    //Check authorization on the requested object. Convert the DN to PD format
(/app/func/etc)
    auth.PDCheckAuthorizationPAC(creds, "/IF Menu" + formattedDN, "r");
        if (menuLogger.isInfoEnabled()) menuLogger.info("authorization checked!!!!!!!");
} catch (com.lmfs.framework.security.PDAuthz.PDAuthzUnauthorizedException unauth) {
    if (menuLogger.isInfoEnabled()) menuLogger.info("The authorization failed for " + "/IF
Menu" + com.lmfs.framework.util.StringExt.getPDFormattedDN(item.dn, pdMenuBase));
    //unauth.printStackTrace();
    isAuthorized = false;
} catch (Exception e) {
    e.printStackTrace();
    isAuthorized = false;
}
if (isAuthorized){
    result.addElement(item);
} else {
    isAuthorized = true;
}

```

Figure 25: Example of PDCheckAuthorizationPAC taken from the Menu.java File of the Menu System Test Component

Table 27: com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl

com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl	
public PDServletAuthzImpl ()	Default Constructor.
public PDServletAuthzImpl (javax.servlet.http.HttpServletRequest)	
<ul style="list-style-type: none"> In param – Servlet request with which to use credential information. 	Default Constructor.
public getAuthorizedObjects (java.lang.String[], java.lang.String)	
<ul style="list-style-type: none"> In Object java.lang.String-Objects being checked for access. In Perm java.lang.String-Permissions requested. 	Returns a list of authorized objects in a new string array.
private getPDAuthnInfo ()	
<ul style="list-style-type: none"> Param javax.servlet.http.HttpServletRequest-Servlet request with which to use credential information. 	Returns identity information related to this authorization object.
public PDCheckAuthorization (java.lang.String, java.lang.String)	

com.lmfs.framework.security.PDAuthz.Servlet.PDServletAuthzImpl	
<ul style="list-style-type: none"> • Param javax.servlet.http.HttpServletRequest-Servlet request with which to use credential information. • Object java.lang.String-Object for which to check access. • Perm java.lang.String-Permissions desired on object being checked. 	Check authorization based on set identity information.
private setPDAuthnInfo (com.lmfs.framework.security.PDAuthn.Servlet.PDServletAuthnInfoImpl)	
<ul style="list-style-type: none"> • NewPDAuthn com.lmfs.framework.security.PDAuthn.PDAuthnImpl 	Sets identity information related to this authorization object.

3.5 com.lmfs.framework.util.*

This package provides general-purpose utility classes. It was originally designed on and used in a Windows NT/x86 environment, and was successfully tested on Solaris 2.7.

The classes provided in this package are:

Table 28: com.lmfs.framework.util Class

com.lmfs.framework.util.StringExt	A class that does not extend String, but contains one method- getPDFormattedDN , which converts an LDAP DN to a name in Policy Director format.
com.lmfs.framework.util.ArrayExt	A class that does not extend Array, but contains methods to convert array data.
com.lmfs.framework.util.PropertiesExt	A class that translates a variety of data formats into the Properties structure.

3.5.1 com.lmfs.framework.util.StringExt

StringExt is an extension of functionality to **java.lang.String**, specifically created for DN-to-Policy Director Object Name parsing. The class is also available for other future generic and miscellaneous IF-related string manipulation routines.

Table 29: com.lmfs.framework.util.StringExt

com.lmfs.framework.util.StringExt	
public StringExt ()	Default Constructor.
public static getPDFormattedDN (java.lang.String, java.lang.String)	
<ul style="list-style-type: none"> • In BaseDN - portion of the DN that should be ignored when creating the Policy Director-Formatted name. 	Returns a DN in reverse format, delimited with '\s, and without the BaseDN. Example: DN

com.lmfs.framework.util.StringExt	"cn=Mike,ou=MyOrgUnit,o=MyOrg,c=us" as parameter 1 passed with DN Base as "o=MyOrg,c=us", would return \MyOrgUnit\Mike.
public static main (java.lang.String[])	
• In args - Unused in this class.	A command-line test method.

3.5.2 com.lmfs.framework.util.ArrayExt

ArrayExt is a *helper* class to extend the functionality of **java.util.Array**.

The usage is method-specific, as the helper class does not actually extend the array class. See each method for usage information.

Table 30: com.lmfs.framework.util.ArrayExt

com.lmfs.framework.util.ArrayExt	
Public ArrayExt ()	Default Constructor.
Public static arrayToString (java.lang.Object[])	<ul style="list-style-type: none"> In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method.
•	Appends all <code>.toString()</code> elements of the array directly after one another in a string, and returns that string.
Public static arrayToString (java.lang.Object[], java.lang.String)	<ul style="list-style-type: none"> In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method. In string - the suffix to append to each element in the object array.
•	Appends suffix (parameter 2) to all the <code>.toString()</code> elements of the array directly after one another in a string, and returns that string.
Public static arrayToString (java.lang.String, java.lang.Object[])	<ul style="list-style-type: none"> In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method In string - the prefix to insert in front of each element in the object array.
•	Prefixes prefix (parameter 1) to all the <code>.toString()</code> elements of the array directly after one another in a string, and returns that string.
Public static arrayToString (java.lang.String, java.lang.Object[], java.lang.String)	<ul style="list-style-type: none"> In string - the prefix to insert in front of each element in the object array. In Object - an array of objects. Assumed to be strings or instantiations of classes that have the <code>toString</code> method. In string - the suffix to append to each element in the object array.
•	Prefixes prefix (parameter 1) and appends suffix (parameter 2) to all the <code>.toString()</code> elements of the array directly after one another in a string, and returns that string.
Private static doWork (java.lang.String, java.lang.String[], java.lang.String)	Does the actual work of converting array elements to strings, and prefixing or appending desired prefixes or suffixes.
public static main (java.lang.String[])	
• Args java.lang.String[]	A unit test method.

3.5.3 com.lmfs.framework.util.PropertiesExt

PropertiesExt is an extension of functionality to **java.util.Properties**. It is used to translate a variety of data formats into the Properties structure. See each constructor for usage details.

The following code shows an example of the use of the **com.lmfs.framework.util.PropertiesExt** class.

```
•••  
public boolean readGlobalRefreshValue() {  
    menuServletLogger.info("Reading refresh.properties file...");  
  
    // Use the classpath to find and load the Refresh.properties  
    java.util.ResourceBundle bundle = null;  
    try {  
        bundle = (java.util.PropertyResourceBundle)  
java.util.PropertyResourceBundle.getBundle("MenuServlet");  
        } catch (java.util.MissingResourceException mrb) {  
            menuServletLogger.error("Missing Resource Exception opening Refresh.properties  
file:\n" +  
                "Be sure the file exists in the classpath, and that all necessary\n" +  
                "settings exist in the properties file.\n" + mrb);  
    }  
    com.lmfs.framework.util.PropertiesExt pe = new  
com.lmfs.framework.util.PropertiesExt(bundle);  
    return (pe.getProperty("REFRESH") == "TRUE");  
}
```

Figure 26: Example of PropertiesExt from the MenuServlet.java file of the Menu System Test Component

Table 31: com.lmfs.framework.util.PropertiesExt

com.lmfs.framework.util.PropertiesExt	
public PropertiesExt ()	Calls default parent constructor.
public PropertiesExt (java.lang.String)	
<ul style="list-style-type: none">In string - the key to read and parse.	Constructs from a string. The parameter is a string to read in and parse, the delimiter is ". (Thus format is "key=value key=value key=value")
public PropertiesExt (java.lang.String, java.lang.String)	
<ul style="list-style-type: none">In string - a list of key/value pairs, that are delimited by parameter 2.In string - the delimiter that separates the key/value pairs in parameter 1.	Constructs from two strings. The first parameter is string to parse and read in, the second parameter is a major delimiter (delimiter around key/value pairs).
public PropertiesExt (java.util.Properties)	
<ul style="list-style-type: none">In props - a Properties object, this should be copied into the <i>PropertiesExt</i> object.	Calls parent constructor with default properties set.
public PropertiesExt (java.util.ResourceBundle)	Constructs a properties object from a ResourceBundle .
public PropertiesExt (java.util.ResourceBundle, java.lang.String)	
<ul style="list-style-type: none">In rb - a resource bundle to read into this <i>PropertiesExt</i> object.	Constructs a properties object from a ResourceBundle . If ResourceBundle is null, constructs ResourceBundle

com.lmfs.framework.util.PropertiesExt	
• In string - a filename that indicates that should indicate the Resource Bundle to load into this <i>PropertiesExt</i> .	from a file named in parameter two.
public test ()	
	A test method.

3.6 SessionInfo Structures

The **SessionInfo** structure provides a common, extendable security data structure for passing required session information around the GCSS-AF system. In the Framework Architecture, WebSphere Advanced Edition Servlets combine context information from the Menu Servlet with security information from Policy Director WebSeal, and place that combined information in a **SessionInfo** structure. The Servlets then pass that information to Component Broker servers to enable method-level authorization checks.

IDL for Security Structures

The **SessionInfo** is defined in the **IFCBSecurityInfo.IFCBSecurityInfoAuthz** class file located in the **IFCBSecurityInfoC.jar** file and in the C++ **IFCBSecurityInfoFile.hh** header file, and in the CORBA **IFCBSecurityInfo** Model.

The following types are defined: **DynInfoType**, **UserSecInfoStruct**, **AppSessionInfoStruct**, **UserSessionInfoStruct**, and **SessionInfoStruct**.

DynInfoType-DynInfoType is a dynamic field to allow extensibility without having to redefine the structure and force applications to recompile

```
typedef sequence<string> DynInfoType;
```

UserSecInfoStruct-UserSecInfoStruct includes the user parameters required to make an authorization decision.

```
struct UserSecInfoStruct {  
    string username;  
    string groups;  
    string creds;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct UserSecInfoStruct
```

Figure 27: Example of UserSecInfoStruct Code

AppSessionInfoStruct-AppSessionInfoStruct includes parameters to aid the application in identifying its location in the Policy Director Object Space.

```
struct AppSessionInfoStruct {  
    string baseNameCtx;  
    string baseNameQualifier;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct AppSessionInfoStruct
```

Figure 28: Example of AppSessionInfoStruct Code

UserSessionInfoStruct-UserSessionInfoStruct is a structure to allow additional non-security related information to be captured and passed around about the user.

```
struct UserSessionInfoStruct {  
    IFCBSecurityInfo::UserSeclInfoStruct userSeclInfoData;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct UserSessionInfoStruct
```

Figure 29: Example of UserSessionInfoStruct Code

SessionInfoStruct-SessionInfoStruct is combines the Application and User Information about a session in a single structure to pass as a parameter amongst applications (Servlets, EJBs, and CORBA components)

```
struct SessionInfoStruct {  
    IFCBSecurityInfo::UserSessionInfoStruct userSessionInfoData;  
    IFCBSecurityInfo::AppSessionInfoStruct appSessionInfoData;  
    IFCBSecurityInfo::DynInfoType dynInfo;  
}; // end struct SessionInfoStruct
```

Figure 30: Example of SessionInfoStruct Code

3.7 IFCBSecurityInfo

The **IFCBSecurityInfo** Model is a Integration Framework Component that should be used by EJB's and CORBA objects for authenticating an application if necessary (end user's are authenticated by WebSEAL). It is also used for making authorization decisions. Table 32: IFCBSecurityInfo provides a description of the methods available in the IFCBSecurityInfo Model. A description of their use is provided below. For additional information, refer to Section 5.3.2.4 of the GCSS-AF Developer's Guide.

Table 32: IFCBSecurityInfo

IFCBSecurityInfo::IFCBSecurityInfoAuthz	
<pre>public boolean gcssafDecisionAccessAllowed(in IFCBSecurityInfo::UserSessionInfoStruct inUserSessionInfoData,in string inSecLabelString,in string inPermission,in string appPDLLabel,in long inCacheTimeout)</pre>	
<ul style="list-style-type: none"> ◆ In IFCBSecurityInfo::UserSessionInfoStruct In UserSessionInfoData – structure identifying the user including the user's name, groups the user belongs to, and the user's credential. The user's credential is the only piece that is used to make the actual authorization decision. The other attributes of the structure are available for audit purposes. ◆ In string inSecLabelString – string that identifies the object within the Policy Director Object Namespace. This may be a fully qualified reference within the Namespace or a relative reference. Relative references are prepended with the appPDLLabel to create a fully qualified reference within the PD Object Namespace. ◆ In string inPermission – a single character that identifies the permission, the action, to be checked ◆ In string appPDLLabel – a string that identifies the upper levels of the Policy Director Namespace for this request. It is only used when the inSecLabelString is a relative reference ◆ In long inCacheTimeout – a time value in seconds that identifies how long a permission in cache would be considered valid. This parameter is currently ignored a caching mechanism has not currently been implemented. Constants of NOCACHE, DEFAULTCACHE, and MAXCACHE are specified in IFCBSecurityInfo 	Evaluates whether a given user is authorized to perform a specific action (permission) on a specific object. The method returns true if the user is authorized.
<pre>public string gcssafGetAuthorizedObjects(in IFCBSecurityInfo::UserSessionInfoStruct userSessionInfoData,in string inListOfSecLabels,in string inPermission,in string appPDLLabel,in long inCacheTimeout)</pre>	<ul style="list-style-type: none"> ◆ In IFCBSecurityInfo::UserSessionInfoStruct In UserSessionInfoData – structure identifying the user including the user's name, groups the user belongs to, and the user's credential. The user's credential is the only piece that is used to make the actual authorization decision. The other attributes of the structure are available for audit <p>Evaluates whether a given user is authorized to perform a specific action (permission) on a set of objects. The method returns a list of objects that the user is authorized to use.</p> <p>This method is not currently implemented.</p>

IFCBSecurityInfo::IFCBSecurityInfoAuthz	
<p>purposes.</p> <ul style="list-style-type: none"> ◆ In string inListOfSecLabels – string that identifies the object within the Policy Director Object Namespace. This may be a fully qualified reference within the Namespace or a relative reference. Relative references are prepended with the appPDLLabel to create a fully qualified reference within the PD Object Namespace. ◆ In string inPermission – a single character that identifies the permission, the action, to be checked ◆ In string appPDLLabel – a string that identifies the upper levels of the Policy Director Namespace for this request. It is only used when the inSecLabelString is a relative reference • In long inCacheTimeout – a time value in seconds that identifies how long a permission in cache would be considered valid. This parameter is currently ignored a caching mechanism has not currently been implemented. Constants of NOCACHE, DEFAULTCACHE, and MAXCACHE are specified in IFCBSecurityInfo 	
<code>public string gcssafGetClientAuditData();</code>	
	Returns the audit data for the most recent authorization check performed by a given security object.
<code>public IFCBSecurityInfo::SessionInfoStruct gcssafSetSessionInfo(in IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo,in string baseNameQualifier,in string baseNameCtx,in IFCBSecurityInfo::DynInfoType appDynInfo,in IFCBSecurityInfo::DynInfoType sessionDynInfo);</code>	<p>Used to facilitate construction and initialization of a SessionInfo structure. Typically used after issuing a gcssafAuthenticateUser for building the rest of a SessionInfoStruct for passing to additional methods.</p>

IFCBSecurityInfo::IFCBSecurityInfoAuthz	
used to supply additional information about a specific session	
Public boolean gcssafAuthenticateUser(in string userDn,in string password,inout IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo)	
<ul style="list-style-type: none"> ◆ In string userDn – The x.500 Distinguished Name of the user from the Policy Director user registry ◆ In string password – the password that matches the password attribute of the Distinguished Name in the PD user registry. ◆ Inout IFCBSecurityInfo::UserSessionInfoStruct userSessionInfo 	Returns a boolean flag to indicate whether a user is authenticated. It also outputs a UserSessionInfo struct containing the user's identity information (username, groups the user belongs to, and the credentials of the user).

3.7.1.1 Authentication

The GCSS-AF Integration Framework relieves the MA from providing a separate authentication mechanism. The MA has the responsibility for trusting that WebSEAL has properly authenticated a user and accepts the identity that is passed to it as the originating identity for all actions. The MA will use this identity for Access Control and Audit requirements.

The MA has the responsibility for defining component identities in the scenarios identified in Table 33: MA Authentication Scenarios. This is not meant to be an exhaustive list, but merely to stimulate the MA to think about how access control and the authentication mechanisms as part of its design task.

Table 33: MA Authentication Scenarios

Situation	Example
Components that are not invoked directly by an end user may need to authenticate to the system as a component identity.	<p>For example a trigger application that reads and processes messages off of a queue may not have access to the originating user information in a form that would allow it to make specific authorization decisions based on that originating user. The methods that the application is invoking require an identity to make its access control decisions.</p> <p>Another example is a wrapped legacy application where a file is FTP'd into a directory. An MA can process the file and invoke secured methods using the component's identity.</p>

Situation	Example
Components that need to act on behalf of a user to invoke methods that the user would not normally be given carte blanche access to may need to authenticate to the system as a component identity.	A Finance MA may provide a method to checkForSufficientFunds on an account based on an account number and a dollar amount. The finance MA doesn't want to open this method up to all end users as this may allow unauthorized users to gather information about other accounts. The finance MA creates another method called userCheckForSufficientFunds that verifies that the user has access to that account before providing feedback. Users are allowed access to this method and a component identity is created that is allowed access to the original checkForSufficientFunds method.

The IF provides the **gcssafAuthenticateUser** method that is part of the **IFCBSecurityInfoAuthz Object** in Component Broker to perform this function for the application. The **sessionInfoStruct** created by this function can be passed as a parameter to CORBA and EJB methods just as the **sessionInfoStruct** created for the user and passed in by the Servlet or EJB. To make applications work properly, it may be necessary to add additional parameters, as required by the invoked application. For example, the *location* and *basenamequalifier* parameters of the **sessionInfoStruct** are required by the test components.

```
com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl UUIDKey = new
com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl();
UUIDKey.generate();
static private org.omg.CORBA.Object obj=null;
obj = theHome.createFromPrimaryKeyString(UUIDKey.getUuid());
x = IFCBSecurityInfoAuthzHelper.narrow(obj);
IFCBSecurityInfoStructObjectHelper usi = new IFCBSecurityInfoStructObjectHelper();
UserSessionInfoStructHolder h = new UserSessionInfoStructHolder();
h.value = usi.sessionInfoStructData.userSessionInfoData;
booleanResult = x.gcssafAuthenticateUser("cn= ILS-BASE1-
PDC.TRIGMON,ou=USAF,ou=PKI,ou=DoD,o=U.S. Government,c=us",
"AppsPassword", h );
```

Figure 31: IFCBSecurityInfoStruct

Typically this would be followed by a call to **gcssafSetSessionInfo** for creating a **SessionInfoStruct** that can be passed around as the input parameter to other CORBA and EJB methods.

3.7.1.1.2 Providing Access Control Checks in MA Software

The tasks required to perform this Step:

- Accept the **sessionInfoStruct** type as a parameter to the method
- Build the parameters required by the **gcssafAccessDecisionAllowed** method

- C. Find/Create a **CBSecurityInfo** model instance (scope host-scope-widened)
- D. Invoke the **gcsafAccessDecisionAllowed** method
- E. Turning security on in SMUI

Step A -Accept the sessionInfoStruct type as a parameter to the method

Each method that needs to be protected with access control checks shall accept a **sessionInfoStruct** structure as a parameter. If the method does not directly make access control checks, but invokes other methods that do require access control checks, then the method would still need to accept a **sessionInfoStruct** structure to be able to pass it to the invoked methods. The sections below describe how to perform this for CORBA and EJB Server Components.

CORBA Server Components -See Figure 32: Example IDL from PDCSessionModule of the PDC Test Component provides an example IDL for a CORBA Server Component. CORBA Server Components need to create a dependency on the IFCBSecurityInfo model in the Component Broker Object Builder tool when any of the methods use the **sessionInfoStruct** type. This dependency is reflected in the IDL by having the **IFCBSecurityInfoFile.idl** include file.

```
...
#include <IFCBSecurityInfoFile.idl>

...
module PDCSessionModule {
    interface PDCSessionAO;

    interface PDCSessionAO : IManagedClient::IManageable
    {
        ...
        ...
        void addPart(in long pdcID,in PDCHelperModule::PartRecord pRecord,in
IFCBSecurityInfo::SessionInfoStruct sessInfoStruct ) raises
        (IManagedClient::IDuplicateKey,PDCHelperModule::PDCException);
        ...
    }; // end interface PDCSessionAO
}; // end of module PDCSessionModule
```

Figure 32: Example IDL from PDCSessionModule of the PDC Test Component²

EJB Server Components -See Figure 33: Example IDL from com_lmfs_framework_testcomponents_requisition_OrderingTie.java of the Requisition Component Test Component provides an example IDL for an EJB Server Component. EJB Server Components need to have the **IFSservices.jar** file in their classpath in order to use the **sessionInfoStruct** type.

² Snippet taken from PDCSessionClasses.idl

```
...
public interface com_lmfs_framework_testcomponents_requisition_OrderingTie extends java.rmi.Remote
{
...
...
// Bean-specific business methods
    public java.lang.String placeAnOrder_object_(IFCBSecurityInfo.SessionInfoStruct arg0,
java.lang.String arg1, java.lang.String arg2, java.lang.String arg3, int arg4, java.lang.String arg5) throws
java.rmi.RemoteException, java.lang.Exception;
...
}
```

Figure 33: Example IDL from com_lmfs_framework_testcomponents_requisition_OrderingTie.java of the Requisition Component Test Component

Step B -Build the parameters required by the gcssafAccessDecisionAllowed method

To build the parameters required by the Access Control method, the MA Developer has to use information available from the code itself and information obtained and coordinated with the Operations and Support Organization.

The parameters of the *gcssafAccessDecisionAllowed* method are:

Table 34: Parameters of the GCSSAF/AccessDecisionAllowed Method

Parameter	How Obtained
UserSessionInfoData	Obtained from sessionInfoStruct parameter. This structure is created and passed in from the invoking application (Servlet, JSP, other CORBA or EJB Server Component) as part of the sessionInfoStruct parameter.
SecLabelString	The Policy Director Object Space (relative or fully qualified context). If this field is formatted as a relative context, then it is pre-pended with the AppPdLabel field to define the Policy Director Object space to check the permission against. Additional information for determining the contents of this field is provided below
Permission	Action that the method is trying to perform.
AppPdLabel	Initial context. Built with information from the sessionInfoStruct parameter and MA specified information. Additional information for determining the contents of this field is provided below.
CacheTimeout	Should be set to 0 for now. The design was anticipating a caching mechanism in the future.

UserSessionInfoData -The **UserSessionInfoData** parameter is contained in the **sessionInfoStruct** that is passed as a parameter to applications that require security access control checks. The data for the **userSessionInfoData** piece of the **sessionInfoStruct** is created by Policy Director as a part of the authentication process.

SecLabelString -The purpose of the *SecLabelString* and the *AppPdLabel* in the *gcssafAccessDecisionAllowed* method is to identify the object as it is represented in Policy Director.

To create **SecLabelString**, the MA Developer can determine the names of the module, interface, and method of the method they are trying to protect. In the code example, the format that the Access Control method is expecting this information is as follows: **String secLabelStr = "[./PDCSessionModule/PDCSessionAO/addPart/]"**; Because it was prepended with a “.” it is a relative Policy Director Namespace Context and the **AppPdLabel** parameter will be used to fully qualify it.

AppPdLabel -To create the **AppPdLabel**, the MA Developer obtains certain information from the **sessionInfoStruct** parameter. The **sessionInfoStruct** parameter contains the **baseNameQualifier** and **baseName** context. The **baseNameQualifier** parameter identifies the initial levels of the Policy Director Object Namespace where this application resides. It is composed of the fixed /GCSS-AFAPPS root context for GCSS-AF applications and the functional domain. For our pseudo-supply test application, the *baseNameQualifier* is **/GCSS-AFAPPS/ILS**.

The **baseNameCtx** is the location that the data represents. Examples from the IF Test Components for this field are: ENTERPRISE, BASE1, BASE2, and BASE3.

It is incumbent on the invoking app to determine this information and supply

Appended to the end of the **AppPdLabel** is the Application Identity. As identified in **Step 4 - Determine Level and Type of Protection** of this process, this label is created by the MA and coordinated with the Operations and Support Organization.

The design of the application may determine an alternate approach of obtaining the **AppPdLabel** parameter. The recommendation is to limit the amount of hard coding, except for static content.

```
public void addPart( int pdcID, PDCHelperModule.PartRecord pRecord,  
IFCBSecurityInfo.SessionInfoStruct sessInfoStruct)  
throws com.ibm.IManagedClient.IDuplicateKey,  
PDCHelperModule.PDCEException  
{  
    ...  
    // local variables  
    // security Label String  
    String secLabelStr = "[./PDCSessionModule/PDCSessionAO/addPart/]" ;  
    // appPDLLabel String  
    String appPDLLabelStr = sessInfoStruct.appSessionInfoData.baseNameQualifier +  
    "/" + sessInfoStruct.appSessionInfoData.baseNameCtx +  
    "/PDC" ;  
    ...  
}
```

Figure 34: Example of Obtaining Parameters Required by the gcssAccessDecisionAllowed Method

Step C -Find/Create an IFCBSecurityInfo model instance (scope server-scope-widened)

Figure 35: Code Example of Finding IFCBSecInfo Home and Figure 36: Code Example of Creating IFCBSecurityInfo Instance provide an example of the code necessary to create and locate an **IFCBSecurityInfo** model instance.

```
private com.ibm.IManagedClient.IHome getIFCBSecInfoHome()

{
    /**
     * This method will locate the Home for the IFCBSecInfo class
     * It will use a server-scope-widened server based on the current
     * serverName to locate that home.
    */

    ...
    // local variables
    //      temp CORBA Object
    org.omg.CORBA.Object obj = null ;

    try {
        if (iIFCBSecInfoHome == null ) {
            // getSSWFactFinder() returns a name service with the following parameters
            // "host/resources/factory-finders/" +
            // CBSeriesGlobal.serverName() + "-server-scope-widened"
            obj = getSSWFactFinder().
                find_factory_from_string("IFCBSecurityInfo::IFCBSecurityInfoAuthz.object interface") ;
            ...
            iIFCBSecInfoHome = com.ibm.IManagedClient.IHomeHelper.narrow(obj) ;
            ...
        } // end if
    }
    catch (Exception exc) {
        logCat.error("PDCSessionAO-" + location() + "::getIFCBSecInfoHome::Exception caught: ", exc) ;
    }
    ...
    return iIFCBSecInfoHome ;
}
}
```

Figure 35: Code Example of Finding IFCBSecInfo Home

```
private IFCBSecurityInfo.IFCBSecurityInfoAuthz createIFCBSeclnfoAuthz()

{
    ...
    /**
     * This method creates the IFCBSecurityInfoAuthz object using the home
     * @return IFCBSecurityInfo.IFCBSecurityInfoAuthz
     */
    if ( iFCbSecInfAthz == null ) {

        ...
        try {
            // create and generate the uuid key
            com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl iFCbSecKey = new
            com.ibm.IBOIMExtLocal._IUUIDPrimaryKeyImpl();
            iFCbSecKey.generate();

            org.omg.CORBA.Object obj = getIFCBSeclnfoHome().createFromPrimaryKeyString(
                iFCbSecKey.getUuid() );
            iFCbSeclnfoAthz = IFCBSecurityInfo.IFCBSecurityInfoAuthzHelper.narrow( obj );
        }
        catch (Exception exc) {
            logCat.error("PDCSessionAO- " + location() + "::createIFCBSeclnfoAuthz::Exception caught: ", exc);
        }
        ...
    } // end if

    return iFCbSeclnfoAthz;
    ...
}
```

Figure 36: Code Example of Creating IFCBSecurityInfo Instance

Step D -Invoke the gcsafAccessDecisionAllowed method

Refer to Figure 37: Code Example of Invoking the gcsafAccessDecisionAllowed Method as an example for invoking the *gcssafAccessDecisionAllowed* method.

```
public void addPart( int pdclD, PDCHelperModule.PartRecord pRecord,
IFCBSecurityInfo.SessionInfoStruct sessInfoStruct)

    throws com.ibm.IManagedClient.IDuplicateKey,
PDCHelperModule.PDCEception

{

    ...

    // check if user is allowed to access the method

    boolean accessAllowed = createIFCBSecInfoAuthz().gcssafDecisionAccessAllowed

( sessInfoStruct.userSessionInfoData,
secLabelStr,
"K",
appPDLLabelStr,
0 ) ;

    ...

if ( accessAllowed ) {

    try {

// Perform the action

    ...

}

    catch (com.ibm.IManagedClient.INoObjectWKey nowk) {

// used as an example catch for the code snippet, other exceptions are handled as well

    ...

    }

    ...

} // end if accessAllowed

    else {

// User not authorized. Perform error handling

        logCat.warn("PDCSessionAO- " + location() + "::addPart::ACCESS DENIED!!") ;

        throw new PDCHelperModule.PDCEception( "PDCSessionAO- " + location() + "::addPart::ACCESS DENIED!!" ) ;

    }

    ...

}

}
```

Figure 37: Code Example of Invoking the gcsafAccessDecisionAllowed Method

The method **gcssafGetClientAuditData** returns the audit data for the most recent authorization check performed by a given security object. This method works when each client obtains its own reference to a different security object, so the audit data storage is unique per client. Thus, the behavior of **gcssafGetClientAuditData** from an object with multiple or static references is undefined. The string returned contains the following data if available: the username, the user's credentials, the object for which access was checked, the permission desired on the object, and the result of the decision.